TUGAS AKHIR – KI141502

RANCANG BANGUN *E-LEARNING* PEMROGRAMAN PADA MODUL *STUDENT FEEDBACK SYSTEM*

RACHMANIA ILAVI
NRP 5112100168

Dosen Pembimbing I
Rizky Januar Akbar, S.Kom, M.Eng

Dosen Pembimbing II
Abdul Munif, S.Kom, M.Sc.

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

DESIGN AND IMPLEMENTATION OF STUDENT FEEDBACK MODULE ON E-LEARNING PLATFORM FOR PROGRAMMING

RACHMANIA ILAVI
NRP 5112100168

First Advisor
Rizky Januar Akbar, S.Kom, M.Eng

Second Advisor
Abdul Munif, S.Kom, M.Sc.

Department of Informatics
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN *E-LEARNING* PEMROGRAMAN PADA MODUL *STUDENT FEEDBACK SYSTEM*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma dan Pemrograman
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
RACHMANIA ILAVI
NRP: 5112100168

Disetujui oleh Pembimbing Tugas Akhir:

1. Rizky Januar Akbar, S.Kom, M.Kom. .....
   (NIP. 198701032014041001) (Pembimbing 1)

2. Abdul Munif, S.Kom, M.Sc. .....
   (NIP. 198608232015041004) (Pembimbing 2)

SURABAYA
JUNI, 2016

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN *E-LEARNING* PEMROGRAMAN PADA MODUL *STUDENT FEEDBACK SYSTEM*

Nama Mahasiswa : RACHMANIA ILAVI
NRP : 5112100168
Jurusan : Teknik Informatika FTIF-ITS
Dosen Pembimbing 1 : Rizky Januar Akbar, S.Kom, M.Eng
Dosen Pembimbing 2 : Abdul Munif, S.Kom, M.Sc

Abstrak

E-Learning bisa diartikan sebagai pembelajaran yang dilakukan di media elektronik baik secara formal maupun informal. Di Jurusan Teknik Informatika, E-Learning digunakan peserta didik sebagai sarana pengumpulan tugas yang diberikan oleh dosen. Dosen seringkali meminta peserta didik membuat kode program yang deskriptif atau instruktif sesuai dengan soal yang diminta dosen. Tetapi pada kenyataannya, banyak peserta didik yang menulis kode program kurang sesuai dengan soal yang diminta, padahal penjelasan soal sudah sangat jelas. Sehingga dibutuhkan sebuah E-Learning Pemrograman yang dapat digunakan peserta didik maupun dosen untuk melaksanakan pembelajaran pemrograman. Pada E-Learning pemrograman diharapkan mampu menyediakan sarana agar dosen dapat menentukan kode program yang dibuat oleh peserta didik ini sesuai dengan struktur jawaban dosen. Selain itu, diharapkan mampu memberikan feedback berupa teks kemiripan dua kode program.

Tugas akhir ini mengimplementasikan metode Smith-Waterman untuk mendeteksi kemiripan dua buah kode sumber. Kode sumber diubah menjadi sebuah AST dengan menggunakan parser ANTLR. AST tersebut lalu diubah menjadi sequence yang menjadi inputan metode Smith-Waterman.

Pada tugas akhir ini hasil yang didapat untuk menilai kemiripan dua buah source code menggunakan metode Smith-

Waterman menghasilkan akurasi sebesar 93,54% dengan menggunakan total 71 source code peserta didik dan satu source code dosen dalam tiga kali uji coba dengan dataset yang berbeda. Sedangkan, metode Smith-Waterman yang dimodifikasi menghasilkan akurasi sebesar 98.9% dengan menggunakan data yang sama dengan metode Smith-Waterman.
Kata kunci: Smith-Waterman, AST, ANTLR, C++, E-Learning, feedback, similarity

DESIGN AND IMPLEMENTATION OF STUDENT FEEDBACK MODULE ON E-LEARNING PLATFORM FOR PROGRAMMING

Student's Name	**: RACHMANIA ILAVI**
Student's ID	**: 5112100168**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Rizky Januar Akbar, S.Kom, M.Eng**
Second Advisor	**: Abdul Munif, S.Kom, M.Sc**

Abstract

E-Learning can be defined as learning that is done in electronic media, both formally and informally. In the Department of Informatics Engineering, E-Learning can be used by students as a place to submit assignments. Teachers ask student to create source code which is descriptively same as the question. In fact, a lot of students write souce code different from what question wants. Therefore it needs an E-Learning platform that can be used for both students and teachers to implement programming. Teachers can determine student's source code is structurally same as teacher's source code by using E-Learning. Beside that, it can give feedback such as a text that contains similarity of two source code file.

This theses implements Smith-Waterman method to detect similarity between two source code files. Those source code files are converted to AST using ANTLR parser. Then ASTs are converted to sequence to be Smith-Waterman's input.

In this theses result, to determine similarity of two source code files using Smith-Waterman method results an acuracy of 93,54% by using 71 source code files and one teacher's source code file in three trials with different dataset. Meanwhile, Smith-Waterman modification method results an acuracy of 98.9% which uses the same data as Smith-Waterman method.

Keywords : Smith-Waterman, AST, ANTLR, C++, E-Learning, feedback, similarity

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah Swt yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**RANCANG BANGUN *E-LEARNING* PEMROGRAMAN PADA MODUL *STUDENT FEEDBACK SYSTEM"***

yang merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada :

1. Bapak Sugeng Hery Sucipto dan Ibu Suliarsih selaku orang tua penulis yang selalu memberikan dukungan doa, moral, dan material yang tak terhingga kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Rizky Januar Akbar, S.Kom, M.Eng dan Bapak Abdul Munif, S.Kom, M.Sc selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Bapak Hudan Studiawan, S.Kom, M.Kom. selaku dosen wali penulis yang telah memberikan arahan, masukan dan motivasi kepada penulis
4. Bapak Darlis Herumurti, S.Kom, M.Kom. selaku kepala jurusan Teknin Informatika ITS dan segenap dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
5. Sahabat penulis yang tidak dapat disebutkan satu per satu yang selalu membantu, menghibur, menjadi tempat bertukar ilmu dan berjuang bersama-sama penulis.

6. Teman-teman penulis mahasiswa Teknik Informatika angkatan 2012 yang menjadi menjadi tempat bertukar ilmu, pengalaman, dan berjuang bersama-sama dari tahun 2012 hingga sekarang.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Juni 2016

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

E-Learning merupakan pembelajaran jarak jauh yang memudahkan peserta didik dalam menuntut ilmu karna tidak terikat ruang dan waktu, atau bisa diartikan sebagai pembelajaran yang dilakukan di media elektronik baik secara formal maupun informal. Di Jurusan Teknik Informatika, *E-Learning* digunakan peserta didik sebagai sarana pengumpulan tugas yang diberikan oleh dosen. Selain itu, peserta dapat mengunduh materi-materi yang diunggah oleh dosen. Membuat program adalah hal yang sering diberikan oleh dosen kepada peserta didik di jurusan Teknik Informatika. Sehingga dibutuhkan sebuah *E-Learning* Pemrograman yang dapat digunakan peserta didik maupun dosen untuk melaksanakan pembelajaran pemrograman.

Pada bidang teknologi informasi telah banyak dikembangkan beberapa aplikasi pendeteksian kemiripan kode program. Salah satu caranya adalah menggunakan AST (*Abstract Syntax Tree*) sebagai data yang akan diolah [1]. AST didapatkan dari hasil penguraian kode program berbahasa C++ menggunakan *generator ANTLR*. Dari AST akan dicari berapa nilai kesamaan dua buah kode program dengan menggunakan algoritma Smith-Waterman [2].

Dosen seringkali meminta peserta didik membuat kode program yang deskriptif atau instruktif sesuai dengan soal yang diminta dosen. Jawaban peserta didik harus sesuai dengan soal yang diminta, tidak boleh lebih atau kurang. Dosen menginginkan peserta didik untuk terbiasa membuat program. Tetapi pada kenyataannya, banyak peserta didik yang menulis kode program kurang sesuai dengan soal yang diminta, padahal penjelasan soal sudah sangat jelas.

Pada *E-Learning* pemrograman diharapkan mampu menyediakan sarana agar peserta didik dapat membuat kode program dan dosen dapat menentukan kode program yang dibuat

oleh peserta didik ini sesuai dengan struktur jawaban dosen. Selain itu, diharapkan mampu memberikan *feedback* berupa teks kemiripan dua kode program.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut.

1. Membangun *back-end* sistem *E-Learning* untuk menilai kemiripan dua buah source code C++ berdasarkan strukturnya;
2. Membangun *back-end* sistem *E-Learning* yang mampu memberikan *feedback* berupa bagian mana pada dua buah kode sumber yang mirip.

1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Aplikasi *back-end E-Learning* untuk pengecekan dua buah kode sumber dan pemberian *feedback* potongan sumber kode yang mirip dibuat dengan menggunakan bahasa pemrograman Java.
2. Dataset *source code* yang digunakan sebagai input menggunakan bahasa C++ dan difokuskan pada paradigma pemrograman berorientasi objek.
3. Dataset yang digunakan adalah *source code* tugas mahasiswa pada kelas PBO C tahun ajaran 2013/2014.
4. Bentuk *Sequence Tree* untuk setiap kode sumber dibuat dengan menggunakan *listener* pada *parser* ANTLR.
5. Metode pengecekan dua buah kode sumber yang digunakan adalah Smith-Waterman dan Smith-Waterman yang telah dimodifikasi.
6. Aplikasi ditujukan untuk dosen pemrograman yang ada di Teknik Informatika.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Menghitung nilai kemiripan *source code* milik dosen dan *source code* milik peserta didik.
2. Memberikan *feedback* berupa potongan kode pada dua buah kode sumber yang mirip.

1.5 Manfaat

Manfaat dari tugas akhir ini sebagai berikut:

1. Mempermudah dosen dalam memantau dan menghitung nilai tugas yang diberikan untuk peserta didik.
2. Mempermudah dosen dalam mengetahui potongan kode sumber peserta didik yang memiliki kemiripan dengan kode sumber dosen.
3. Bagi penulis, tugas akhir ini sebagai sarana untuk mengimplementasikan ilmu di bangku kuliah agar memiliki nilai guna bagi masyarakat terutama untuk membantu sekolah dalam bidang teknologi.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal

Tahap awal tugas akhir ini adalah menyusun proposal tugas akhir. Pada proposal, diajukan gagasan untuk menghitung kemiripan dua buah kode sumber milik dosen dan peserta didik, serta mendapatkan *feedback* berupa potongan kode sumber yang mirip.

1.6.2 Studi Literatur

Pada tahap ini dilakukan untuk mencari informasi dan studi literatur apa saja yang dapat dijadikan referensi untuk membantu pengerjaan tugas akhir ini. Informasi didapatkan dari buku dan literatur yang berhubungan dengan metode yang digunakan. Informasi yang dicari adalah mengubah kode program menggunakan *parser* ANTLR menjadi AST (*Abstract Syntax Tree)*, mengubah hasil AST ke dalam bentuk *sequence* tanpa mengubah struktur kode program yang asli, metode Smith-Waterman yang telah dimodifikasi guna menghitung kemiripan dua buah *source code* berorientasi objek.

1.6.3 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal tugas akhir. Untuk membangun aplikasi *back-end* pada modul *student feedback system* ini menggunakan bahasa java yang diimplementasikan menggunakan *Eclipse Luna* sebagai IDE, ANTLR *plugin* sebagai kakas bantu untuk memparsing kode program menjadi bentuk AST, Notepad untuk membaca potongan kode hasil kemiripan dua buah kode sumber, serta *Object Relational Database Management System* PostgreSQL untuk menyimpan data *Sequence* hasil *parsing* dan derajat kemiripan antar kode program.

1.6.4 Pengujian dan Evaluasi

Pada tahap ini algoritma yang telah disusun diuji coba dengan menggunakan data uji coba yang ada. Data uji coba tersebut diuji coba dengan tujuan mengetahui kemampuan metode yang dipakai dan mengevaluasi hasil tugas akhir.

1.6.5 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencangkup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori dari metode, algoritma, serta penjelasan kakas bantu pendukung yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang ANTLR *parser,* AST(*Abstract Syntax Tree),* Algoritma Smith-Waterman, bahasa pemrograman Java dan PostgreSQL.
3. Bab III. Perancangan Perangkat Lunak
   Bab ini berisi pembahasan mengenai perancangan dari pembuatan Tugas Akhir seperti pembentukan AST menggunakan bantuan *Listener* yang ada pada *parser* ANTLR dan penggunaan metode Smith-Waterman yang dimodifikasi.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk kode sumber dari metode Smith-Waterman yang dimodifikasi, dan penggunaan *Listener*.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dari metode Smith-Waterman yang dimodifikasi. Uji coba dilakukan dengan membandingkan kemiripan kode sumber dosen dan peserta didik dan didapatkan 3 hasil, Mirip, Agak Mirip, dan Tidak Mirip.

6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan kode sumber program secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi dua bagian, yaitu pembahasan mengenai teori-teori dasar yang digunakan serta beberapa kakas yang mendukung pembuatan tugas akhir. Teori-teori tersebut diantaranya adalah *Abstract Syntax Tree* (AST), Algoritma Smith-Waterman. Sedangkan kakas yang digunakan diantaranya adalah *ANTL,* Java, PostgreSQL, dan beberapa kakas lain yang mendukung pembuatan tugas akhir.

2.1. *E-Learning*

E-Learning dapat didefinisikan sebagai sebuah bentuk teknologi informasi yang diterapkan di bidang pendidikan berupa website yang dapat diakses di mana saja [3]. Selain itu, *E-Learning* merupakan pembelajaran jarak jauh yang memudahkan peserta didik dalam menuntut ilmu karna tidak terikat ruang dan waktu, atau bisa diartikan sebagai pembelajaran yang dilakukan di media elektronik baik secara formal maupun informal.

2.2. Feedback

Feedback dapat didefinisikan sebagai pemberian sebuah umpan balik kepada seseorang untuk mempermudah komunikasi. *Feedback* pada bidang Teknologi Informasi bisa digunakan untuk pemberitahuan atau *alert* bahwa sebuah sistem berjalan dengan baik dan dapat digunakan juga untuk mendukung hasil atau *output* dari suatu sistem.

2.3. Kakas Pendukung Pembuatan Tugas Akhir

Beberapa kakas bantu pendukung yang digunakan dalam pembuatan tugas akhir ini adalah sebagai berikut:

2.3.1. ANTLR

ANTLR (*ANother Tool for Language Recognition*) adalah sebuah *parser generator* yang *powerful* untuk membaca,

memproses, dan mengeksekusi, atau mengartikan struktur teks atau file binari. Dari *grammar* (aturan bahasa), ANTLR menghasilkan sebuah *parser* yang dapat membangun *parser trees* atau yang disebut *Abstract Syntax Tree (AST)* [4].

ANTLR menyediakan alat bantu untuk mekanisme pembacaan AST yang ada pada *runtime library* ANTLR, yaitu *listener* dan *visitor*. Baik Mekanisme *tree-walking* pada *listener* dan *visitor* dilakukan secara (*Depth First Search* (DFS). Tetapi pada *Listener* mempunyai dua fungsi startDocument() dan endDocument(), sedangkan visitor tidak [5] .

2.3.2. PostgreSQL

PostgreSQL adalah sebuah sistem basis data yang disebarluaskan secara bebas menurut Perjanjian lisensi BSD. Piranti lunak ini merupakan salah satu basis data yang paling banyak digunakan saat ini, selain MySQL dan Oracle. PostgreSQL menyediakan fitur yang berguna untuk replikasi basis data. Fitur-fitur yang disediakan oleh PostgreSQL antara lain *complex queries, foreign keys, triggers, updatable views, transactional integrity,* dan *multiversion concurrency control* [6].

2.4. Dasar Teori yang Digunakan

Dasar teori yang digunakan dalam tugas akhir ini adalah sebagai berikut:

2.4.1. Algoritma Smith-Waterman

Algoritma Smith-Waterman dapat digunakan untuk menghitung kemiripan dua buah dokumen, tak terkecuali sebuah kode program. Algoritma ini digunakan terutama dalam bidang biologi komputasional, algoritma ini mempunyai efek yang bagus dalam pencocokan yang optimal. Pertama, algoritma ini akan menggunakan matriks, lalu menghasilkan sub matriks yang berisi semua kemungkinan kesamaan nilai dengan metode iteratif, membandingkan nilai-nilai dari sub matriks hingga didapatkan nilai yang optimal (didapatkan nilai tertinggi) [2].

Untuk dua buah *AST Sequence*: $S = s_1, s_2, s_3, \dots, s_n$ dan $T = t_1, t_2, t_3, \dots, t_m$, menurut algoritma *dinamic programming,* dibutuhkan sebuah matriks berukuran $(n+1) \times (m+1)$, penghitungan matriks dan sub matriks dapat menggunakan rumus seperti Persamaan 2.1 dan Persamaan 2.2 dan untuk algoritma penghitungan sub matriks *D* dapat diliat pada Gambar 2.1. Nilai indeks *i* dan *j* adalah $1 \le i \le n \; dan \; 1 \le j \le m$.

1. Kondisi awal

$$D(i,0) = D(0,j) = 0. \tag{2.1}$$

2. Relasi rekursi

$$D(i,j) = max \begin{cases} 0, \\ D(i-1, j-1) + f(i,j), \\ \max\{D(i-1,j) - W_x\}, \\ \max\{D(i,j-1) - W_y\}, \end{cases} \tag{2.2}$$

```
i = LengthS, j = LengthT
FOR i > 0 AND j > 0
   IF (D[i,j ]= D[i-1, j-1] + f(i,j) THEN
       i--
       j--
   Else IF (D[i,j] = D[i-1, j] + Wx)  THEN
       i--
   Else if (D[i,j] = D[i-1, j-1] + Wy) THEN
       j--;
```

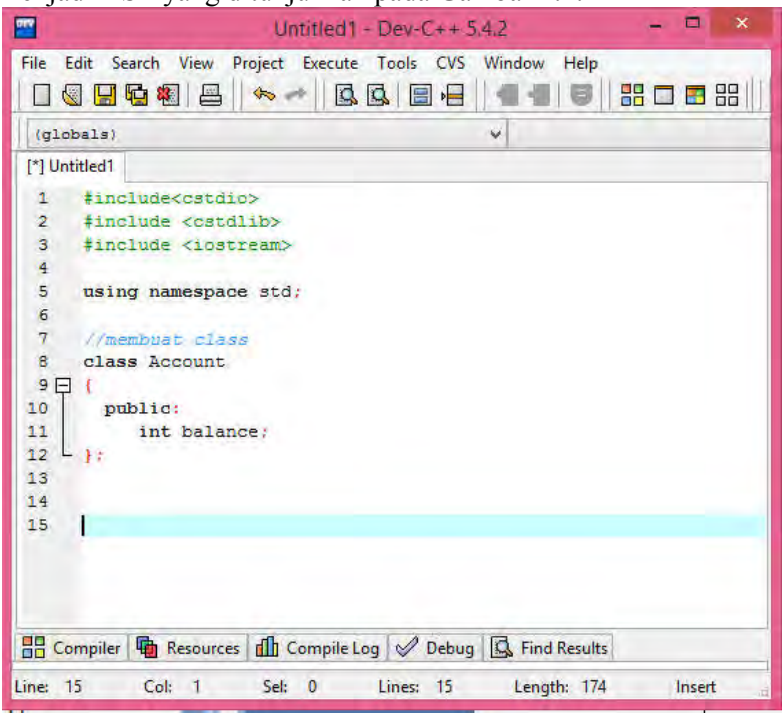
Kode Sumber 2.1 Algoritma Smith-Waterman untuk penghitungan sub matriks

Untuk $1 \le i \le n$ dan $1 \le j \le m, W_x \; W_x$ adalah kolom *penalty* dengan panjang x yang ditambahkan pada *sequence*, sedangkan W_y adalah kolom *penalty* dengan panjang y yang ditambahkan pada *sequence,* dan *f(i,j)* bernilai *match* atau *missmatch*. D adalah matriks dua dimensi berisi hasil penghitungan algoritma Smith-Waterman.

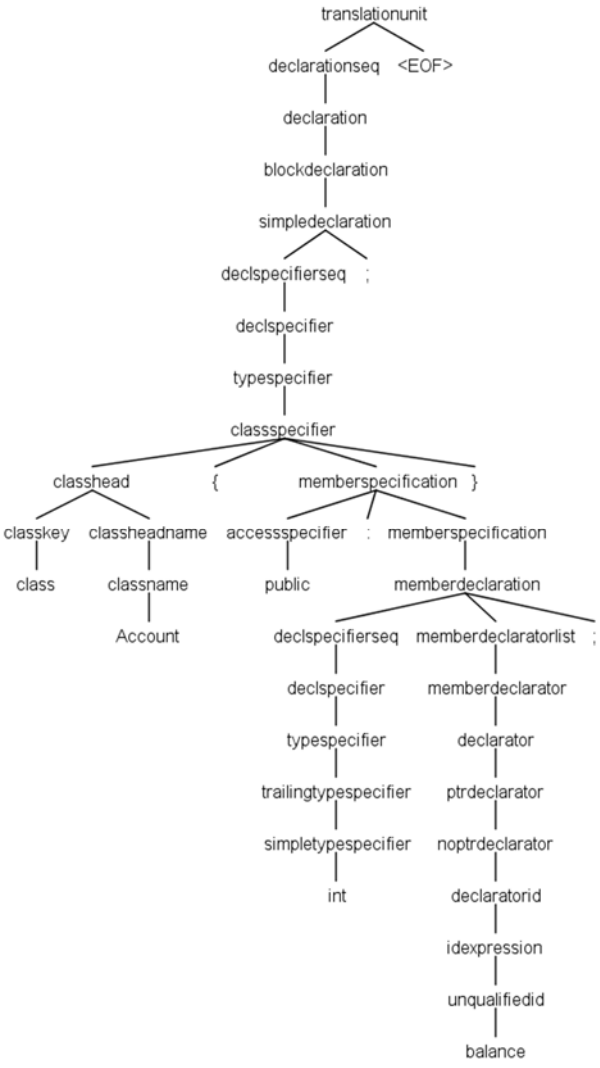
2.4.2. Abstract Syntax Tree (AST)

AST adalah sebuah pohon yang merepresentasikan struktur dari sebuah kode program dengan bahasa pemrograman tertetu. Dalam aplikasi ini, AST diperoleh dari *tree parser* yang dihasilkan oleh ANTLR. Kode program akan diubah secara otomatis oleh ANTLR ke dalam bentuk pohon atau sering disebut juga AST. Dari AST inilah yang akan diproses lebih lanjut untuk menghitung tingkat kemiripan kode program peserta didik dan jawaban dosen.

Gambar 2.1 merupakan contoh kode program C++ yang selanjutnya akan di*parsing* dengan menggunakan ANTLR menjadi AST yang ditunjukkan pada Gambar 2.2.



Gambar 2.1 Contoh Kode Program C++ Sederhana



Gambar 2.2 Hasil AST Kode Sumber 2.1

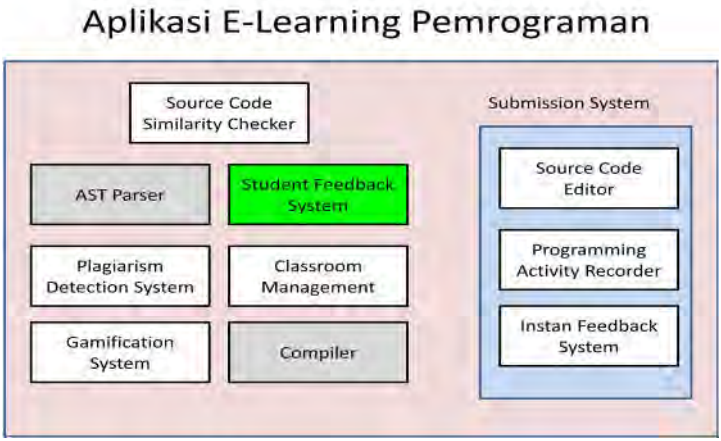
(Haplaman ini sengaja dikosongkan)

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai perancangan dan pembuatan sistem perangkat lunak serta arsitektur sistem *E-Learning* pemrograman. Perancangan yang dibuat pada tugas akhir ini adalah *preprocessing* (praproses) atau pembentukan *AST* dengan mengolah data hasil dengan menggunakan *listener* yang terdapat pada *parser* ANTLR, dan membandingkan dua kode sumber menggunakan metode *Smith-Waterman* yang telah dimodifikasi, serta hasil kemiripan dua buah kode sumber tersebut.

3.1 Arsitektur *E-Learning* Pemrograman secara Umum

Secara keseluruhan Aplikasi E-Learning Pemrograman mempunyai arsitektur yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Arsitektur E-Learning Pemrograman

Keterangan Gambar 3.1:

- : modul yang akan diimplementasikan penulis pada tugas akhir ini.
- : modul yang tidak diimplementasikan karena ada library yang tersedia
- : modul yang tidak diimplementasikan penulis.

Modul Student Feedback System: Modul untuk melakukan penilaian atau *grading* kemiripan kode sumber dosen dan peserta didik, serta pemberian saran / *feedback* berupa potongan kode yang mirip.

3.2 Modul Student Feedback System

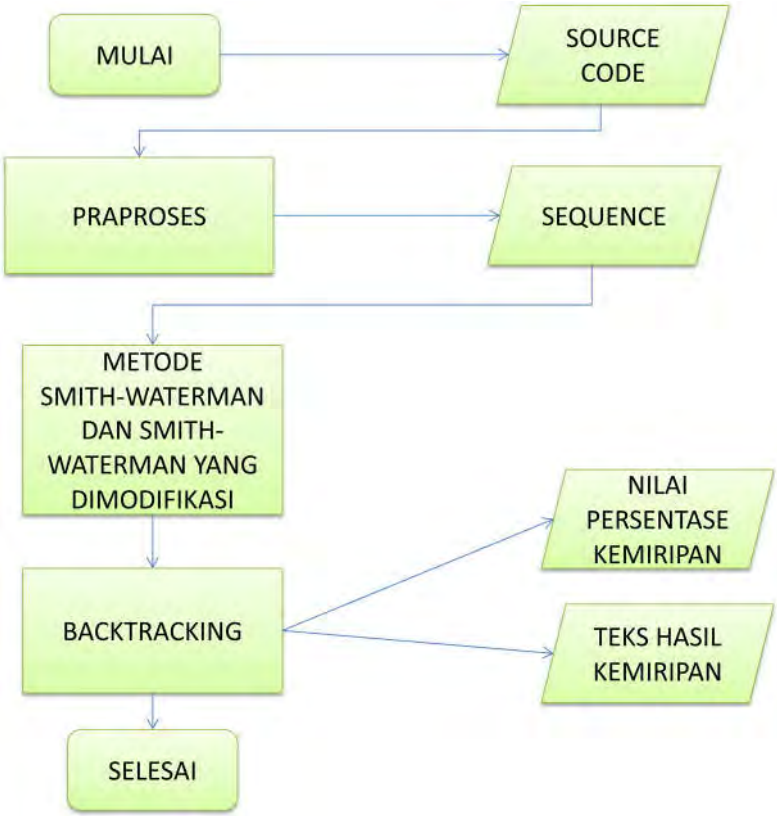
Perancangan *E-Learning* pada modul *Student Feedback System* akan dijelaskan sebagai berikut:

3.2.1 Desain Umum Sistem

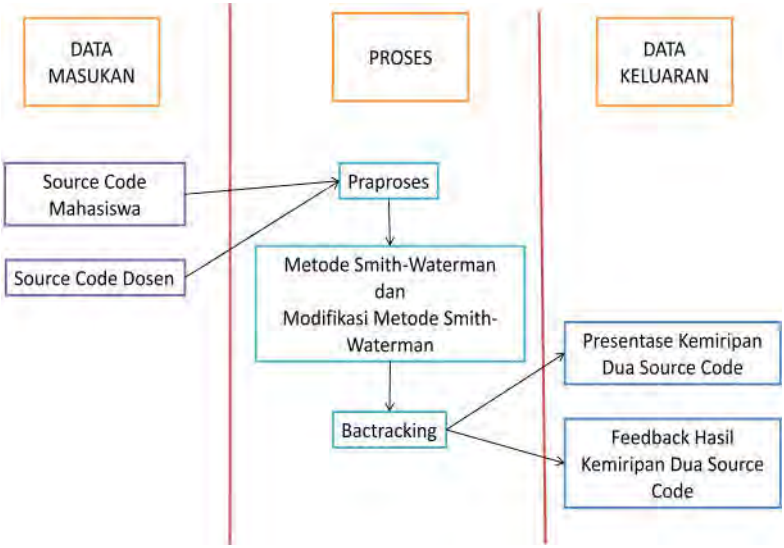
Rancangan *back-end* perangkat lunak untuk menilai kemiripan dua buah *source code* dan pemberian *feedback* berupa teks yang menunjukkan hasil kode sumber yang mirip dimulai dengan tahap praproses yaitu mengolah dan mengubah data *source code* menjadi sebuah *sequence* atau *fingerprint*. Tahap praproses menggunakan bantuan *parser* ANTLR dengan menggunakan fungsi *listener* yang telah dimodifikasi menjadikan *source code* ke dalam bentuk *sequence*. Bentuk *sequence* ini merepresentasikan bentuk *AST* dari *source code*.

Tahap selanjutnya adalah membandingkan antara dua buah *sequence* dari kode sumber dosen dan peserta didik kemudian menghitung tingkat kemiripannya menggunakan metode *Smith-Waterman* yang telah dimodifikasi. Hasil dari proses membandingkan dua buah kode sumber ini adalah nilai *similarity* (nilai kemiripan antar dua buah kode sumber) dan sebuah teks yang berisi kode yang sama antara dua buah *source code*.

Diagram alir penilaian dan pemberian *feedback* yang ditunjukkan pada Gambar 3.2 dan gambaran data masukan, proses, data keluaran ditunjukkan pada Gambar 3.3.



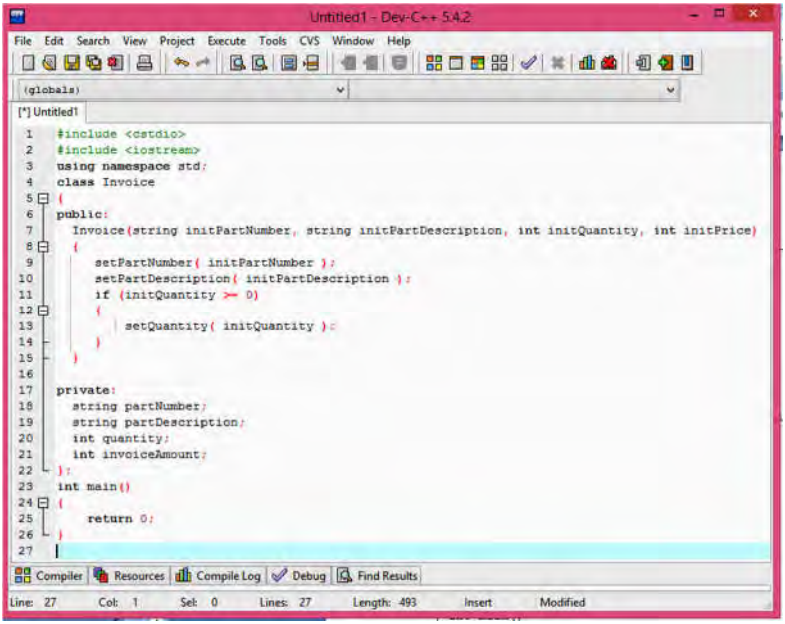
Gambar 3.2 Alir Penilaian dan Pemberian Feedback



Gambar 3.3 Gambaran Data Masukan, Proses, Data Keluaran

3.2.2 Data Masukan

Data masukan adalah data yang digunakan sebagai masukan awal dari sistem. Data yang digunakan adalah kode sumber peserta didik Pemrograman Berorientasi Objek (PBO) Tahun Ajaran 2013/2014 kelas C. Data terdiri dari 31 buah kode sumber peserta didik dan 1 kode sumber dosen. Data ini selanjutnya akan diproses pada Tugas Akhir ini. Contoh data mentah ditunjukkan pada Gambar 3.4.



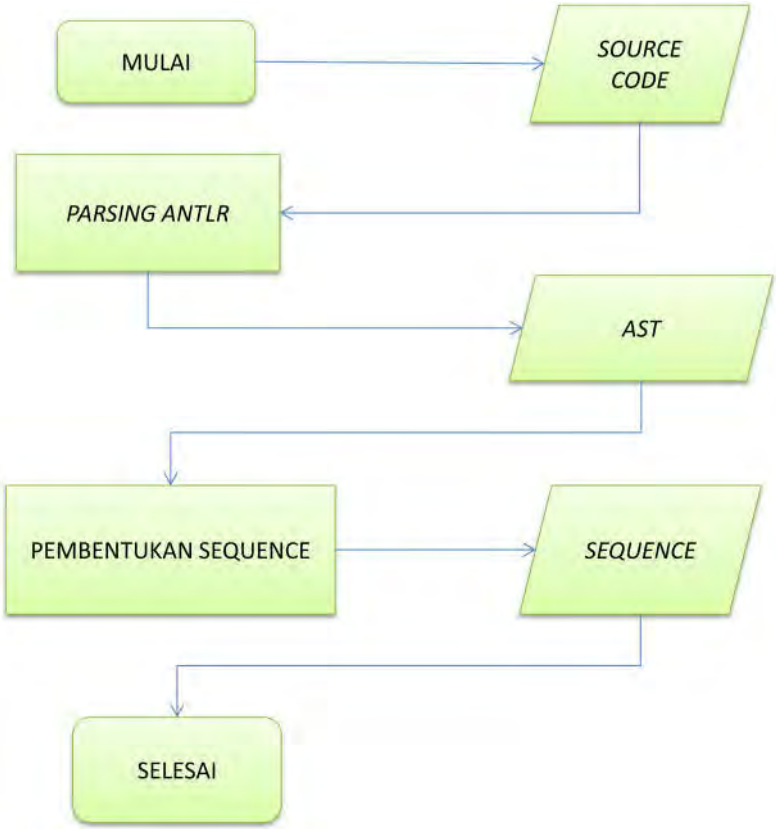
Gambar 3.4 Contoh Data Masukan Source Code Dosen dan Peserta Didik

3.2.3 Proses

Pada sub bab ini akan dijelaskan mengenai proses dari pengolahan data masukan hingga menjadi data keluaran. Serta dijelaskan metode yang digunakan dalam pengolahan data.

3.2.3.1 Praproses

Praproses merupakan cara untuk mengubah data menjadi bentuk yang cocok untuk proses penghitungan kemiripan, yaitu mengubah kode sumber menjadi *sequence.* Langkah praproses digambarkan pada diagram pada Gambar 3.5 dan Ilustrasi perubahan *source code* menjadi AST menjadi *sequence* dapat dilihat pada Gambar 3.6.



Gambar 3.5 Diagram Alir Praprose

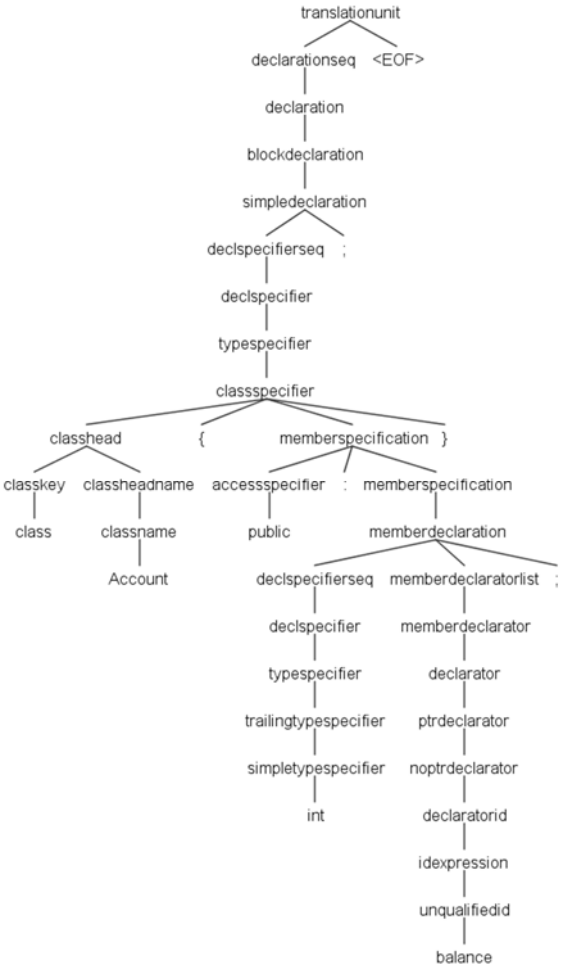
Gambar 3.6 Ilustrasi pengubahan source code menjadi sequence

3.2.3.1.1 Parsing ANTLR

Data masukan dari proses praproses ini adalah *source code C++* dosen maupun peserta didik. Kode sumber tersebut di*parsing* menggunakan *plugin* ANTLR menjadi *Abstract Syntax Tree* (AST) kemudian dibaca dan diolah menggunakan fungsi listener yang terdapat pada ANTLR *parser*. Contoh AST dapat dilihat pada Gambar 3.7.

3.2.3.1.2 Pembentukan *Sequence*

Pembentukan sequence memerlukan modifikasi fungsi listener yang terdapat pada ANTLR. Listener digambarkan sebagai suatu *tree-walker* yang berjalan-jalan mengunjungi setiap node dari *AST*. Listener memiliki dua fungsi utama yaitu *Enter* dan *Exit*. *Enter* digunakan ketika *tree-walker* tersebut mengunjungi suatu *node* dan exit digunakan ketika *tree-walker* tersebut selesai mengunjungi suatu *node*. Isi *node* tersebut didapatkan dari sebuah *grammar* C++ yang digunakan ANTLR untuk mem*parsing* kode program, sehingga menghasilkan data hasil parser berupa kumpulan *node* atau AST *(Abstract Syntax Tree).*



Gambar 3.7 Hasil AST pada Gambar 3.6

Tidak semua segmen *source code* akan diambil pada tugas akhir ini. Karena ini merupakan perhitungan kemiripan *source code* berdasarkan strukturnya, maka hanya beberapa segmen dari kode sumber yang di-*parsing* oleh listener *parser* milik ANTLR. Bagian-bagian tersebut dijelaskan pada Tabel 3.1.

Tabel 3.1 Grammar C++ ANTLR yang Di-*parsing*

No	Rule pada Grammar	Keterangan pada kode sumber
1	Declaration	Sebuah pendeklarasian suatu kode yang umumnya diakhir dengan tanda ";" (titik koma) pada c++
2	Simpletypespecifier	Tipe data pada suatu variabel (*int, char, long, double*, dll)
3	Functiondefinition	Pendeklarasian suatu fungsi (*void , int* , dll)
4	Parametersandqualifiers	Parameter dalam sebuah fungsi
5	Functionbody	Isi suatu fungsi
6	Selectionstatement	Statement *if , else if, else, dan switch* pada c++
7	Iterationstatement	Statement berupa *for, while, do-while* pada c++
8	Condition	Sebuah kondisi didalam statement *if , else if, else, case, while, for, dan do-while* pada c++
9	Classpecifier	Pendeklarasian suatu kelas atau struct pada c++
10	Classkey	Pendeklarasian *key* sebuah kelas atau *struct*
11	Accesspecifier	Jenis *access specifier* pada anggota suatu kelas (public, private, dan protected)
12	Basespecifier	Pendeklarasian sebuah kelas turunan pada c++
13	Assignmentexpression	Pendeklarasian sebuah *assignment* pada program c++
14	Assignmentoperator	Operator yang menghubungkan sebuah *assignment*. Operator terdiri dari (= , += , -= , *= , /= , %=)

15	Unqualifiedid	Penggunaan/deklarasi suatu variabel pada kode program C++
16	Literal	Penggunaan/deklarasi suatu angka pada kode program C++
17	Expressionlist	Isi parameter dari pemanggilan suatu fungsi
18	Postfixexpression	Mendeteksi *statement* yang diikuti dengan operator (. , -> , -- , ++ , dan ( )
19	Shiftexpression	Mendeteksi penggunaan cout << dan cin >> pada kode program C++
20	Jumpstatement	Mendeteksi penggunaan *break*
21	Classname	Mendeteksi nama kelas

Setelah kode sumber tersebut di-*parsing* oleh ANTLR menjadi AST, maka fungsi listener yang dimodifikasi digunakan untuk mengubah bentuk AST tadi menjadi sebuah bentuk *sequence*. Proses modifikasi fungsi listener tersebut adalah dengan merepresentasikan fungsi-fungsi pada grammar hasil parser tadi ke dalam suatu rangkaian huruf dan memberikan *parenthesis* di beberapa huruf untuk menandai bahwa huruf-huruf tersebut berada dalam satu *level tree* atau tidak. Setiap huruf diberi level yang menunjukkan *parent and child* sebuah. Kode berupa rangkaian huruf yang merepresentasikan fungsi-fungsi yang akan diparser pada grammar dijelaskan pada Tabel 3.2.

Tabel 3.2 Pengkodean Rule Grammar C++ menjadi Rangkaian Huruf

No	Rule pada Grammar	Kode huruf
1	Declaration	**A ( )**
2	Simpletypespecifier	**B : tipe** (tipe merupakan tipe variabel, misal B: int)

3	Functiondefinition	**C ( )**
4	Parametersandqualifiers	**D ( )**
5	Functionbody	**E ( )**
6	Selectionstatement	**H : statement** Misal (H : if , H:switch)
7	Iterationstatement	**I : statement ( )** Misal (I : for, I:while, I:do)
8	Condition	**J (kondisi)** (kondisi disini apabila menemui fungsi OR , AND, == , >= , <= , > , < , dan !=)
9	Classspecifier	**K ( )**
10	Accessspecifier	**L : *accessspecifier*** Misal (L : public)
11	Basespecifier	**M**
12	Assignmentexpression	Tidak dikodekan. Hanya digunakan untuk proses praproses
13	Assignmentoperator	Operator yang menghubungkan sebuah *assignment*. Operator terdiri dari **(=)** **(+=)** **(-=)** **(*=)** **(/=)** **(%=)** Operator diikuti oleh suatu variabel (**V**) atau angka (literal)
14	Unqualifiedid	**V** **(**simbol ini dapat diikuti oleh operator dari sebuah assignment, parameter pemanggilan fungsi (**P**) , kondisi (**J**), statement dari

		switch-case (**G**), dan simbol-simbol dari postfix expression.)
15	Literal	Angka yang tertera pada kode program misal : **1** , **2** **(**simbol ini dapat diikuti oleh operator dari sebuah assignment, parameter pemanggilan fungsi (**P**) , kondisi (**J**), statement dari *switch-case* (**G**), dan simbol-simbol dari postfix expression.)
16	Expressionlist	**P** **(**simbol ini dapat diikuti oleh operator dari sebuah assignment, variabel (**V**) , literal (angka), kondisi (**J**), statement dari *switch-case* (**G**), dan simbol-simbol dari postfix expression.)
17	Postfixexpression	**V.** (misal : car.lenght) **V->** (misal : car->lenght) **V.O** (misal : car.move()) **V->O** (misal : car->move()) **V--** (misal : a--) **V++** (misal : a++) **VO** (misal : move())
18	Shiftexpression	Tidak dikodekan. Hanya digunakan untuk proses praproses
19	Jumpstatement	**break** (apabila didalam suatu switc-case menjadi **Gbreak**)
20	Classname	**NK**
21	Classkey	Tidak dikodekan. Hanya digunakan untuk

		mendifinisikan *key* kelas atau *struct*

3.2.3.2 Algoritma Smith-Waterman

Smith-Waterman merupakan metode yang digunakan untuk menghitung nilai kemiripan antara dua buah *sequence* (sekumpulan karakter). Data masukan dalam proses ini adalah dua buah *sequence* yang merupakan hasil dari tahap praproses sebelumnya. Metode *Smith-Waterman* membutuhkan sedikit modifikasi untuk menyelesaikan beberapa *case* dalam pendeteksian kemiripan dua buah kode sumber, karena metode ini hanya mendeteksi secara berurutan tidak dapat mendeteksi kode yang peletakannya dibalik.

Pada metode *Smith-Waterman* ini apabila tiap huruf pada dua buah *sequence* adalah sama dan memiliki level yang sama pula, maka akan disebut *match* yang mempunyai nilai pada perhitungan *Smith-Waterman* tersebut adalah +2. Apabila tidak sama atau *mismatch*, cost nya bernilai -1. Nilai tersebut akan ditambahkan dengan nilai matriks 2 dimensi hasil yang ada pada indeks (*i*-1, *j*-1). Sedangkan ada sebuah nilai yang disebut W (bernilai -1) yang ditambahkan dengan nilai matriks 2 dimensi hasil yang ada pada indeks (*i*-1, *j*) atau (*i, j*-1). Lalu diambil jumlah nilai yang paling besar untuk mengisi matriks hasil serta disimpan arah nya pada matriks *directions,* arahnya disimbolkan untuk atas adalah 3, kiri adalah 2, dan miring adalah 1 jika *match*, dan 9 jika *mismatch*.

Sedangkan modifikasi dilakukan untuk melakukan deteksi terhadap susunan isi kode program yang posisinya ditukar (dibolak-balik). Modifikasi yang dilakukan adalah mencatat semua nilai yang *match* yang akan dimasukkan ke dalam array 1 dimensi. Panjang dimensi adalah baris atau kolom yang paling panjang. Selain itu, indeks nilai *match* akan disimpan pada *arraylist*. Hasil persentase *similarity* menggunakan metode Smith-Waterman yang tidak dimodifikasi maupun yang

dimodifikasi akan dibandingkan dan dicari nilai akurasi yang lebih besar.

3.2.3.3 Backtracking

Backtracking ini digunakan untuk mengetahui kode mana saja yang mirip dari dua buah *source code* dan menghitung nilai kemiripan dua buah *source code.* Baik metode Smith-Waterman dan Smith-Waterman yang dimodifikasi menggunakan rumus yang sama untuk menghitung kemiripan dua buah kode sumber. Rumus menghitung kemiripan dua buah *source code* ditunjukkan pada Persamaan 3.7.

Pada metode *Smith-Waterman*, *backtracking* dimulai dari indeks matriks hasil ke i=panjang baris dan j = panjang kolom. Backtracking mengikuti isi matriks *directions,* jika benilai 2 maka ke kiri, jika bernilai 3 maka ke atas, dan jika bernilai 1 atau 9 maka ke kiri-atas(serong). *Backtracking* berhenti jika indeks i atau j bernilai 0 [7].

Sedangkan *backtracking* modifikasi mencocokkan indeks baris atau kolom dengan isi *arraylist* yang berisi indeks nilai match. Selama indeks *match* belum pernah digunakan, maka indeks tersebut akan dipakai sebagai indeks yang memiliki kemiripan dua buah buah kode sumber.

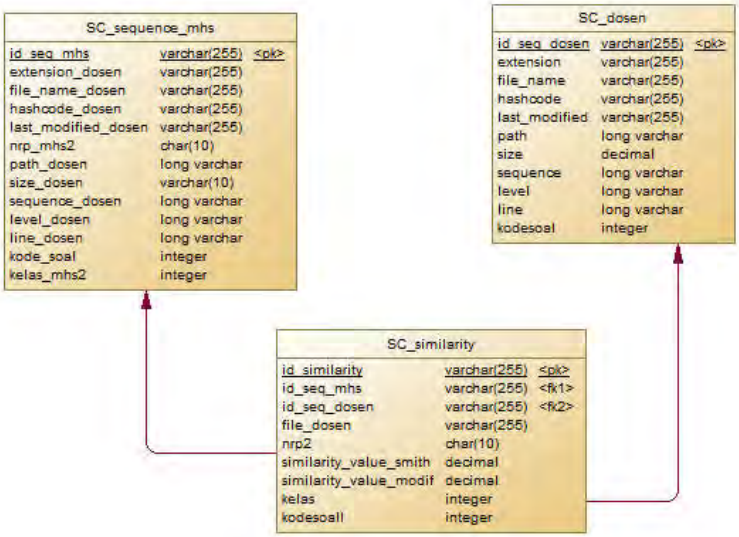
$$\%\text{kemiripan dua kode sumber} = \frac{banyak\ kode\ yang\ mirip}{maks(i,j)} \quad (3.1)$$

3.3 Data Keluaran

Data keluaran yang dihasilkan ada dua. Pertama adalah nilai kemiripan antara kode sumber mahasiswa dan dosen. Kedua adalah *feedback* berupa *highlight* pada teks apa saja yang mirip dari dua buah kode sumber.

3.4 Basis Data

Perancangan basis data pada tugas akhir ini akan dijelaskan pada sub bab ini. Gambar 3.8 menunjukkan PDM *E-learning* pada modul *Student Feedback System.* Sub bab ini berisi penjelasan setiap entitas yang ada pada Gambar 3.8.



Gambar 3.8 PDM E-learning *pada modul* Student Feedback System.

3.4.1 Entitas SC_sequence_mhs

Entitas ini berisi sequence dari hasil *parsing* kode sumber peserta didik yang didapatkan dari proses praproses. Isi entitas ini digunakan sebagai data masukan untuk metode Smith-Waterman dan metode Smith-Waterman yang dimodifikasi Atribut yang terdapat pada entitas ini dapat dilihat di Lampiran pada Tabel A.1.

3.4.2 Entitas sc_dosen

Entitas ini berisi sequence dari hasil *parsing* kode sumber dosen yang didapatkan dari proses praproses. Isi entitas ini digunakan sebagai data masukan untuk metode Smith-Waterman

dan metode Smith-Waterman yang dimodifikasi. Isi entitas ini akan dibandingkan dengan isi entitas sc_sequence_mhs dengan metode tersebut. Atribut yang terdapat pada entitas ini dapat dilihat di Lampiran pada Tabel A.2.

3.4.3 Entitas sc_similarity

Entitas ini berisi nilai kemiripan dua buah kode sequence, yaitu milik dosen dan mahasiswa. Nilai kemiripan didapatkan dari proses *backtracking*. Nilai kemiripan dengan menggunakan metode Smith-Waterman dan Smith-Waterman yang dimodifikasi akan disimpan pada entitas ini. Atribut yang terdapat pada entitas ini dapat dilihat di Lampiran pada Tabel A.3.

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa kode sumber untuk membangun program.

4.1 Lingkungan Implementasi

Implementasi penghitungan kemiripan kode program dosen dan peserta didik menggunakan metode *Smith-Waterman* yang telah dimodifikasi menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i7-3610QM CPU @ 2.30 GHz 2.30 GHz
	Memori	8 GB 1600 MHz DDR3
Perangkat Lunak	Sistem Operasi	Windows 8.1 Pro
	Perangkat Pengembang	Eclipse Luna , ANTLR V4, dan PostgreSQL V9.3

4.2 Implementasi Modul Student Feedback System

Pada sub bab implementasi ini menjelaskan mengenai pembangunan perangkat lunak secara detail dan menampilkan kode sumber yang digunakan mulai tahap praproses hingga penghitungan nilai kemiripan dan hasil kemiripan. Pada tugas akhir ini data yang digunakan seperti yang telah dijelaskan di bab sebelumnya yaitu 31 kode sumber peserta didik dan 1 kode sumber dosen Kelas Pemrograman Berorientasi Objek (PBO). Hal ini dilakukan agar pemrosesan menjadi lebih sederhana.

4.2.1 Implementasi Praproses

Sub bab ini akan menjelaskan beberapa implementasi pada proses praproses, yang akan dijelaskan sebagai berikut:

4.2.1.1 Implementasi *Parsing Source Code* mengunakan ANTLR

Pada tugas akhir ini, *source code* C++ akan di*parsing* menggunakan bantuan ANTLR. S*ource code* dosen dan peserta didik yang akan menjadi data masukan sistem ini. *Source code* akan diubah ke dalam bentuk *AST (Abstrack Syntax Tree)* menggunakan metode *listener* yang ada pada ANTLR. Terdapat grammar C++ yang diambil dari *website github* yang digunakan sebagai aturan pembentukan AST [8]. Kode program C++ dibaca melalui Kode Sumber 4.1 dan pemanggilan metode listener dapat dilihat pada Kode Sumber 4.2.

```
1  read each file in path_folder
2  scan file using grammar C++ lexerparse file
3  using grammar C++ parser
```

Kode Sumber 4.1 Pemanggilan File C++ Peserta Didik

```
1  call root function in grammar C++
2  call treewalker in ANTLR
3  call listener function
4  walk the tree (AST) using treewalker
```

Kode Sumber 4.2 Pemanggilan Fungsi Listener

Pada Kode Sumber 4.1 pemanggilan file dilakukan dengan membaca seluruh file .cpp pada direktori. Untuk kode program dosen dicari nama file yang mengandung kata "dosen". Sedangkan pada kode program peserta didik tidak memiliki kata "dosen".

Pada Kode Sumber 4.2 hal pertama yang dilakukan adalah memanggil metode *listener* pada *ANTLR parser*. Dan menjalankan fungsi *walk* yang ada pada metode *listener*. Fungsi *walk* tersebut berfungsi sebagai *tree* yang akan mengunjungi setiap pagian pada kode program mahasiswa dan menghasilkan sebuah *Abstract Syntax Tree* (AST). AST yang dihasilkan adalah bentuk *tree* yang merupakan representasi dari pemanggilan

metode-metode yang ada pada *grammar C++*. Terdapat pemanggilan variabel *txt* untuk *sequence,* baris kode program, dan level *sequence* untuk setiap *souce code*

4.2.1.2 Implementasi Modifikasi Fungsi *Listener*

Abstract Syntax Tree (AST) yang didapatkan dari Kode Sumber 4.2 sangat banyak dan mempunyai cakupan yang terlalu luas untuk diidentifikasi. Karena itu, tidak semua metode-metode pada *grammar C++* akan digunakan dalam proses penghitungan kemiripan *source code*. Oleh karena itu, *Abstract Syntax Tree* (AST) tersebut akan diubah dengan memodifikasi pemanggilan metode *listener* yang ada pada kelas *MyCppListener* untuk diubah menjadi bentuk *sequence* yang siap disimpan ke dalam basis data. Metode *listener* dimodifikasi agar menghasilkan bentuk *sequence* yang diinginkan. Modifikasi yang dilakukan adalah dengan menyimbolkan hasil *listener* menjadi beberapa kode dan memberikan level pada setiap kode.

Terdapat empat matriks utama yang digunakan, yaitu *AST, sequence, line,* dan *level.* Matriks *AST* digunakan untuk menyimpan bentuk kode pada setiap *sequence*. Matriks *sequence* digunakan untuk menyimpan kode asli *source code* sesuai dengan fungsi-fungsi *listener.* Matriks *line* digunakan untuk menyimpan baris ke berapa fungsi-fungsi *listener* yang dipanggil pada *source code*. Sedangkan matriks *level* digunakan untuk menyimpan kedalaman atau *level tree* setiap *sequence*.

Bentuk modifikasi fungsi *listener* akan dijelaskan lebih detail di bawah ini. Pertama adalah *declaration,* yang disimbolkan dengan "A", dapat dilihat pada kode sumber 4.3. Fungsi ini digunakan sebagai pendefinisian sebuah deklarasi suatu fungsi atau metode (umumnya letaknya di level paling atas dan diakhiri dengan tanda titik koma ";"). Terdapat empat variabel utama yang digunakan, yaitu *sequence_code* (kode untuk *listener* pada *sequence*), *soucecode_content* (kode asli *source code* sesuai dengan fungsi-fungsi *listener), soucecode_line*

(baris ke berapa fungsi-fungsi *listener* yang dipanggil pada *source code*), dan *count_level* (kedalaman atau *level tree* setiap *node*).

Terdapat dua fungsi yaitu `enterDeclaration`, keadaan ketika *listener* mengunjungi sebuah deklarasi dan `exitDeclaration`, keadaan ketika *listener* selesai mengunjungi deklarasi. Jika pada *listener* fungsi `enterDeclaration` dipanggil, akan dilakukan penambahkan simbol "A" pada variabel *sequence_code*, lalu variabel *count_level* dibuat sama dengan 0 saat dimasukkan pada array dan nilai *count_level* ditambah dengan 1. Sedangkan pada keadaan **exitDeclaration,** *count_level* dikurangkan dengan 1. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.1 untuk fungsi **enterDeclaration** dan **exitDeclaration**

```
FUNCTION enterDeclaration(ctx)
 1   append sequence_code to AST
 2   append count_level to level
 3   append sourcecode_line, indeks_start,
     indeks_stop to line
 4   append sourcecode_content to sequence
 6   increase count_level
```

```
FUNCTION  exitDeclaration(ctx)
 1   decrease count_level
```

Kode Sumber 4.3 Fungsi `enterDeclaration` dan `exitDeclaration` pada listener

```
FUNCTION enterClasskey(ctx)
 1   get ctx Text in this function
```

```
FUNCTION enterClassname(ctx)
 1   IF ctx not contains "string" THEN
 2         append sequence_code to AST
 3         append count_level to level
```

```
 4          append sourcecode_line,
 5          indeks_start, indeks_stop to line
 6    END IF
```

```
FUNCTION enterClassspecifier(ctx)
 1    append sequence_code to AST
 2    append count_level to level
 3    append sourcecode_line, indeks_start,
 4    indeks_stop to line
 5    append sourcecode_content to sequence
 6    increase count_level
```

```
FUNCTION exitClassspecifier(ctx)
 1    decrease count_level
```

Kode Sumber 4.4 Fungsi classname, classkey, classpecifier pada listener

Terdapat empat variabel utama yang digunakan, yaitu *sequence_code* (kode untuk *listener* pada *sequence*), *soucecode_content* (kode asli *source code* sesuai dengan fungsi-fungsi *listener), soucecode_line* (baris ke berapa fungsi-fungsi *listener* yang dipanggil pada *source code*), dan *count_level* (kedalaman atau *level tree* setiap *node*).

Kode Sumber 4.4 menjelaskan pendeklarasian suatu kelas menggunakan `enterClasspecifier` dan `exitClasspecifier` pada *listener*. Modifikasi yang dilakukan sama seperti `enterDeclaration` dan `exitDeclaration`, hanya saja disini diberi simbol "K" pada variabel *sequence_code*. Sedangkan fungsi enterClassname digunakan untuk mengambil nama kelas dan nama kelas tidak sama dengan *string* untuk membedakan dengan tipe data "*string*". Karena pada dasarnya, *string* pada C++ adalah suatu kelas. Fungsi `enterclasskey` digunakan untuk mengambil *key* sebuah kelas berupa "class" atau "struct". Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.2 untuk fungsi *classname*, *classkey*, dan *classspecifier*.

```
FUNCTION enterAccessspecifier(ctx)
  1    append sequence_code to AST
  2    append count_level to level
  3    append sourcecode_line, indeks_start,
       indeks_stop to line
  4    append sourcecode_content to sequence
  5    set countAccessspec = 1
  6    increase level_count
```

Kode Sumber 4.5 `enterAccesspecifier` pada listener

Terdapat empat variabel utama yang digunakan, yaitu *sequence_code*(kode untuk *listener* pada *sequence*), *soucecode_content*(kode asli *source code* sesuai dengan fungsi-fungsi *listener), soucecode_line*(baris ke berapa fungsi-fungsi *listener* yang dipanggil pada *source code*), dan *count_level* (kedalaman atau *level tree* setiap *node*). Kode Sumber 4.5 menjelaskan *accesspecifier* dalam suatu kelas menggunakan fungsi `enterAccesspecifier` pada *listener.* Ada 3 jenis *acces specifier* yaitu *public, private,* dan *protected.* Modifikasi yang dilakukan sama seperti `enterDeclaration` dan `exitDeclaration`, hanya saja disini diberi simbol "L:<jenis *accesspecifier*>", contoh : L:public. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.3 untuk fungsi e*nterAccesspecifier* dan *exitAccesspecifier*.

Sedangkan Kode Sumber 4.6 menjelaskan adanya deklarasi turunan sebuah kelas. Tetapi modifikasi yang digunakan pada fungsi `enterBasespecifier` pada *listener* adalah dengan memberikan simbol "M" pada *sequence_code*.

```
FUNCTION enterBasespecifier(ctx)
1  append sequence_code to AST
2  append count_level to level
```

```
3  append sourcecode_line, indeks_start,
4  indeks_stop to line
   append sourcecode_content to sequence
```

Kode Sumber 4.6 Fungsi `enterBasespecifier` pada listener

```
FUNCTION enterSimpletypespecifier(ctx)
1   decrease count_B
2   IF count_B != 1 THEN
3           pop stack st
4   push ctx.getText to stack st
```

Kode Sumber 4.7 Fungsi `enterSimpletypespecifier`

Sedangkan pada Kode Sumber 4.7 menjelaskan deklarasi tipe data pada suatu variabel dengan menggunakan fungsi `enterSimpletypespecifier` pada *listener* untuk mengambil jenis tipe data. Variabel *count_B* adalah penanda jika tipe data digunakan sebelumnya. Tipe sebuah data akan disimpan dalam sebuah *stack* yang akan diolah pada fungsi `enterDeclarator`. Sedangkan fungsi `enterDeclarator` pada *listener* digunakan untuk mengidentifikasikan suatu variabel pada *sequence*. Dengan demikian dapat mendeteksi *int* a, b sebagai dua variabel *int*. Selain itu, digunakan untuk mengidentifikasi suatu *constructor* dan *destructor* pada kelas. Untuk level pada b*asespecifier* dan *declarator* sama dengan level pada simbol sebelumnya yang ada pada *sequence*. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.4 untuk fungsi **`enterDeclarator`** dan **`exitDeclarator.`**

```
FUNCTION enterFunctiondefinition(ctx)
1   append sequence_code to AST
2   append count_level to level
3   append sourcecode_line, indeks_start,
4   indeks_stop to line
5   append sourcecode_content to sequence
6   increase count_level
```

```
FUNCTION exitFunctiondefinition(ctx)
1  decrease count_level
```

Kode Sumber 4.8 Fungsi Functiondefinition pada listener

```
FUNCTION enterParametersandqualifiers(ctx)
1  append sequence_code to AST
2  append count_level to level
3  append sourcecode_line, indeks_start,
   indeks_stop to line
4  append sourcecode_content to sequence
5  count_level += 2
```

```
exitParametersandqualifiers(ctx)
1  count_level -= 2
```

Kode Sumber 4.9 Fungsi Parametersandqualifiers pada listener

Kode Sumber 4.8 menjelaskan pendeklarasian suatu fungsi pada kode program menggunakan fungsi `enterFunctiondefinition` dan `exitFunctiondefinition` pada *listener* ANTRL. Modifikasi yang dilakukan sama dengan modifikasi pada `enterDeclaration` maupun `exitDeclaration` yang sudah dijelaskan sebelumnya hanya saja pada variabel *sequence_code* disimbolkan dengan "C ".

Kode Sumber 4.9 menjelaskan parameter suatu fungsi menggunakan fungsi `enterParametersandqualifiers` dan `exitParametersandqualifiers` pada *listener*. Modifikasi yang dilakukan hampir sama dengan modifikasi pada `enterDeclaration` maupun `exitDeclaration` yang sudah dijelasakan sebelumnya. Hanya berbeda pada perhitungan level, jika sebelumnya level nya di ditambah 1, pada fungsi ini level ditambah 2 dan pada variabel *sequence_code* disimbolkan dengan "D ".

```
FUNCTION enterFunctionbody(ctx)
1  append sequence_code to AST
2  append count_level to level
3  append sourcecode_line, indeks_start,
4  indeks_stop to line
5  append sourcecode_content to sequence
6  count level++
```

```
FUNCTION exitFunctionbody(ctx)
1  count_level--
```

Kode Sumber 4.10 Fungsi Functionbody pada Listener

Kode Sumber 4.10 menjelaskan isi dari suatu fungsi pada kode program menggunakan fungsi `enterFunctionbody` dan `exitFunctionbody` pada *listener*. Modifikasi yang dilakukan sama dengan modifikasi pada `enterDeclaration` maupun `exitDeclaration` yang sudah dijelaskan sebelumnya hanya saja variabel *sequence_code* disimbolkan dengan huruf "C ".

```
FUNCTION enterSelectionstatement(ctx)
1    split ctx by "(" AND store in array
2    IF array contains select_state THEN
3       append selectstat to sequence
5       append count_level to level
6       append sourcecode_line, indeks_start,
7       indeks_stop to line
8       append sequence_code to AST
9    END IF
```

```
exitSelectionstatement(ctx)
1    split ctx by "(" AND store in array
2    append sequence_code to AST
3    IF array contains "if" THEN
5          decrease count_level by 2
6    ELSE IF array[0] contains "switch" THEN
7          decrease count_level by 3
8    END IF
```

Kode Sumber 4.11 Fungsi Selectionstatement pada Listener

Kode Sumber 4.11 menjelaskan penggunaan *if-else* dan *switch* pada kode program menggunakan fungsi `enterSelectionstatement` dan `exitSelectionstatement` pada *listener*. Modifikasi yang dilakukan hampir sama dengan modifikasi pada `enterDeclaration` maupun `exitDeclaration` yang sudah dijelaskan sebelumnya. Namun pada fungsi ini dilakukan pengecekan apakah suatu kode program menggunakan *if-else* atau *switch.* Level pada penggunaan *if-else* ditambah 2, sedangkan level pada penggunaan *Switch-case* ditambah 3. Sedangkan penggunaan *break* pada kode program menggunakan fungsi `enterJumpstatement` pada *listener*. Fungsi *ini* juga digunakan untuk membedakan sequence maupun level pada case satu dengan case lain nya pada penggunaan *switch-case*. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.5 untuk fungsi **`enterSelectionstatement`** dan **`exitSelectionstatement.`**

```
FUNCTION enterIterationstatement(ctx)
1   split ctx by "(" AND store in arraY
2   IF array equals iter_statement THEN
3        append sequence_code to AST
4        append count_level to level
5        append sourcecode_line, indeks_start,
6        indeks_stop to line
7        append sourcecode_content to sequence
8        increase CountLevel
```

```
FUNCTION exitIterationstatement(ctx)
1   set flagIterDowhile = 0
2   CountLevel--
```

Kode Sumber 4.12 Fungsi Iterationstatement pada Listener

Kode Sumber 4.12 menjelaskan penggunaan iterasi pada kode program menggunakan fungsi `enterIterationstatement` dan

`exitIterationstatement` pada *listener.* Variabel *iter_statement* yang dimaksut pada 4.12 adalah **for, while,** dan **do.** Modifikasi yang dilakukan hampir sama dengan modifikasi pada `enterDeclaration` maupun `exitDeclaration` yang sudah dijelasakan sebelumnya. Hanya saja pada fungsi ini dilakukan pengecekan apakah suatu kode program menggunakan *for, while,* atau *do-while.* Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.6 untuk fungsi **`enterIterationstatement`** dan **`exitIterationstatement.`** Untuk penggunaan *iterationstatement* diperlukan perubahan *grammar C++* yang ditunjukkan pada Gambar 4.1. Karena sebelumnya kondisi pada *do while* tidak terdeteksi.

```
iterationstatement
:
    While '(' condition ')' statement
    | Do statement While '(' condition ')' ';'
    | For '(' forinitstatement condition? ';' expression? ')' statement
    | For '(' forrangedeclaration ':' forrangeinitializer ')' statement
;
```

Gambar 4.1 Grammar *Iterationstatement*

Sedangkan kondisi pada *iteration* dan *selection statement* ditunjukkan pada Kode Sumber 4.14 menggunakan `enterCondition` dan `exitCondition` pada *listener.* Pada `enterCondition`, dilakukan pengecekan tiap kondisi apakah mempunyai operator seperti = , !=, ==, >= , <=, > , <, &&, ||, dan adanya operator penjumlahan, pengurangan, perkalian, atau pembagian yang akan disimpan pada variabel *tanda* dan variabel *sequence_code* akan berisi sesuai dengan variabel *tanda.*. Pada `exitCondition` dimodifikasi seperti `exitDeclaration`**.** Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.7 untuk fungsi **`enterCondition`** dan **`exitCondition.`**

```
FUNCTION enterCondition(ctx)
1   set flagCondition = 1
2   increase CountLevel
3   change string ctx.getText to code sequence
4   AND store in condition
5   FOR i = 0 to size of condition
6         IF condition = tanda THEN
7               append sequence_code to AST
8               append count_level to level
9               append sourcecode_line,
10              indeks_start, indeks_stop to
11              line
12              append sourcecode_content to
13              sequence
14        END IF
15  END FOR
```

```
FUNCTION exitCondition(ctx)
1   set flagCondition = 0
2   CountLevel--
```

Kode Sumber 4.13 Fungsi Condition pada listener

Kode Sumber 4.14 mengidentifikasi adanya deklarasi suatu *assignment* melalui fungsi `enterAssignmentexpression` dan `exitAssignmentexpression` pada *listener.* Di dalam `enterAssignmentexpression` akan diambil isi sebuah *assignment*. Lalu diolah atau dimodifikasi jika sebuah *assignment* memiliki sebuah *operator assignment* pada fungsi `enterAssignmentoperator` pada *listener.* Sedangkan pengecekan apakah *assignment* telah benar-benar selesai dikunjungi oleh *tree*, dilakukan pada fungsi `exitAssignmentexpression`**.** Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.8 untuk fungsi `enterAssignmentexpression` dan `exitAssignmentexpression`.

```
FUNCTION enterAssignmentexpression(ctx)
1   set expression = split ctx by "="

FUNCTION exitAssignmentexpression(ctx)
1   IF Assignmentexpression is shiftexpression
    THEN
2       set flag_express = 0
3       decrease count_Level
4   END IF
```

Kode Sumber 4.14 Fungsi Assignment pada listener

Penggunaan suatu *assignment operator* (=, +=, -=, *=, ..., dsb) pada sebuah *assignments expression* dapat dideteksi melalui fungsi `enterAssignmentoperator` pada *listener*. Didalam fungsi ini juga akan diolah untuk mendapatkan kode operasi aritmatika (penjumlahan, pengurangan, perkalian, pembagian, dan modulasi) yang dideklarasikan setelah *assignment operator* ditulis. Selain itu, untuk mengidentifikasi variabel yang dideklarasikan sebelum sebuah *assignment operator* ditulis. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran C.9 untuk fungsi **`enterAssignmentoperator.`**

Adanya sebuah variabel pada suatu kode program dapat diidentifikasi dengan fungsi `enterUnqualifiedid` pada *listener*. Kode sequence yang diberikan kepada variabel adalah huruf **V** yang diikuti dengan suatu simbol atau rangkaian huruf, yang merepresentasikan variabel tersebut merupakan milik suatu *assignment*, kondisi, atau parameter. Variabel juga bisa diikuti oleh suatu postfix. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran C.10 untuk fungsi **`enterUnqualifiedid.`**

Penggunaan angka atau *literal* dapat dideteksi dengan fungsi `enterLiteral` pada *listener*. Variabel diikuti dengan simbol atau rangkaian huruf yang merepresentasikan penggunaan angka atau *literal* tersebut merupakan milik suatu *assignment*, kondisi, maupun parameter. Literal juga bisa dimiliki oleh suatu

string. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran C.11 untuk fungsi **enterLiteral.**

Sebuah parameter pada pemanggilan fungsi diidentifikasi dengan fungsi *Expressionlist* pada listener. Di dalam **enterExpressionlist** akan dihitung jumlah parameter pemanggilan fungsi yang ditunjukkan pada Kode Sumber 4.15. Sedangkan Kode Sumber 4.16 digunakan untuk mendeteksi **cin** ataupun **cout** yang ditandai dengan *flagshift2.*

```
FUNCTION enterExpressionlist(ctx)
1   set flagparametercall++

FUNCTION exitExpressionlist(ctx)
1   set flagParametercall = 0
2   set flagParameterpostfix = 0
```

Kode Sumber 4.15 Fungsi enterExpressionlist pada listener

```
FUNCTION enterShiftexpression(ctx)
1   flagshift++
2   set shiftexpress = ctx.getText
3   IF shifexpress contains "<<" THEN
4         set flagshift2 = 1
5   ELSE IF shifexpress contains ">>" THEN
6         set flagshift2 = 1
```

Kode Sumber 4.16 Fungsi enterExpressionlist pada listener

Kode Sumber 4.17 mengidentifikasi adanya penggunaan *access operator* (.) dan (→), increment (++) dan decrement (--), dan pemanggilan fungsi pada sebuah *source code.* Selain itu pada baris 5-7 mengidentifikasi penggunaan *string* maka nilai variabel *flagString* bernilai 1.

```
FUNCTION enterPostfixexpression(ctx)
1   change string ctx to code sequence AND
2   store in postfixexpress
3   set flagString = 0
4   IF postfixexpress contains string   THEN
5         set flagString = 1
6   END IF
```

Kode Sumber 4.17 Fungsi *enterPostfixexpression* bagian 1

```
1   IF postfix is not string
2         IF (regexPattern2_1 = match) OR
              (regexPattern2_2 = match) THEN
3                append sequence_code to AST
4                append count_level to level
5                append sourcecode_line,
6                indeks_start, indeks_stop to
                 line
                 append ctx.getText to sequence
7         END IF
8   END IF
9
```

Kode Sumber 4.18 Fungsi *enterPostfixexpression* bagian 2

Tabel 4.2 Pattern Regex Bagian 1

Nama Pattern	Pattern
RegexPattern2_1	^(?:[a-zA-Z0-9]{0,61}[a-zA-Z0 9](?:\\((?:\\))))*$
RegexPattern2_2	^(?:[a-zA-Z0-9]{0,61}[a-zA-Z0-9](?:\\((?:[a-zA-Z0-9(),.->+\"']{0,61}(?:[a-zA-Z0-9,>+/*().\"\']{0,61}[a-zA-Z0-9(),.->+\"\'])\\))))*$

Sedangkan untuk mengidentifikasi pemanggilan fungsi digunakan sebuah *regex* (Regular Expression). Pada Kode Sumber 4.18 dan menggunakan *pattern regex* pada Tabel 4.2

akan dideteksi suatu pemanggilan fungsi. Terdapat pengecekkan pemanggilan fungsi yang berada di dalam **case** dan kondisi.

Tabel 4.3 Pattern Regex bagian 2

Nama Pattern	Pattern
regexPattern3_1	^(?:[a-zA-Z0-9\\[\\]]{0,61}[a-zA-Z0-9](?:\\.[a-zA-Z0-9]{0,61}[a-zA-Z0-9])?(?:\\((?:\\))))*$
regexPattern3_2	^(?:[a-zA-Z0-9\\[\\]]{0,61}[a-zA-Z0-9](?:\\.[a-zA-Z0-9]{0,61}[a-zA-Z0-9])?(?:\\((?:[a-zA-Z0-9(),.>+\"']{0,61}(?:[a-zA-Z0-9,->+/*().]{0,61}[a-zA-Z0-9(),.->+\"'])\\))))*$
RegexPattern3_11	^(?:[a-zA-Z0-9\\[\\]]{0,61}[a-zA-Z0-9](?:\\-\\>[a-zA-Z09]{0,61}[a-zA-Z0-9])?(?:\\((?:\\))))*$
RegexPattern3_22	^(?:[a-zA-Z0-9\\[\\]]{0,61}[a-zA-Z0-9](?:\\-\\>[a-zA-Z09]{0,61}[a-zA-Z0-9])?(?:\\((?:[a-zA-Z0-9(),.->+_\"']{0,61}(?:[a-zA-Z0-9,->+/*_().\"']{0,61}[a-zA-Z0-9(),.->+_\"'])\\))))*$
RegexPattern3_111	^(?:[a-zA-Z0-9->.\\[\\]]{0,61}[a-zA-Z0-9](?:\\.[a-zA-Z0-9](?:[a-zA-Z0-9]{0,61}[a-zA-Z0-9]))?(?:\\((?:\\))))*$
RegexPattern3_222	^(?:[a-zA-Z0-9->.\\[\\]]{0,61}[a-zA-Z0 9](?:\\.[a-zA-Z0-9](?:[a-zA-Z0-9]{0,61}[a-zA-Z0-9]))?(?:\\((?:[a-zA-Z0-9(),.->+\"']{0,61}(?:[a-zA-Z0-9,->+/*().]{0,61}[a-zA-Z0-9(),.->+\"'])\\))))*$
RegexPattern3_1111	^(?:[a-zA-Z0-9->.\\[\\]]{0,61}[a-zA-Z0-9](?: \\-\\> [a-zA-Z0-9](?:[a-zA-Z0-9]{0,61}[a-zA-Z0-

	9]))?(?:\\((?:\\)))*$
RegexPattern3_2222	^(?:[a-zA-Z0-9->.\\[\\]]{0,61}[a-zA-Z0-9](?: \\-\\> [a-zA-Z0-9](?:[a-zA-Z0-9]{0,61}[a-zA-Z0-9]))?(?:\\((?:[a-zA-Z0-9(),.->+\"']{0,61}(?:[a-zA-Z0-9,->+/*().]{0,61}[a-zA-Z0-9(),.->+\"'])\\))))*$

Sedangkan untuk mengidentifikasi pemanggilan fungsi digunakan sebuah *regex* (Regular Expression). Kode Sumber 4.19 dengan menggunakan *pattern regex 3_1 dan 3_2* pada tabel 4.3 akan dideteksi adanya penggunaan *access operator* **dot** (.) yang diikuti pemanggilan fungsi. Sedangkan *pattern regex 3_11* dan *3_22* digunakan untuk mendeteksi *access operator* (→). Cara pengidentifikasiannya hampir sama dengan pendeteksian *acess operator* (.), hanya saja pengkodeannya yang berbeda. Jika *access operator* (.) menggunakan "V.O" sedangkan *access operator* (→) menggunakan "V->O".

Pattern regex 3_111 dan 3_222 pada tabel 4.3 digunakan untuk mengidentifikasi *postfix* berupa *access operator* (**.**) yang diikuti oleh access operator lainnya yaitu (**.**) atau (→), lalu diikuti pemanggilan fungsi **( ),** misal a→b**.**eat(). Sedangkan Pattern regex 3_1111 dan 3_2222 pada tabel 4.3 digunakan untuk mengidentifikasi postfix berupa *access operator* (→) yang diikuti oleh access operator lainnya yaitu (**.**) atau (→), lalu diikuti pemanggilan fungsi **( ),** misal a→b**.**eat().

Kode Sumber 4.20 mengidentifikasi adanya pengunaan *access operator* (.) dengan menggunakan *pattern regexPattern11* pada Tabel 4.3. Sedangkan pengidentifikasian *access operator* (→) dengan menggunakan *pattern regexPattern22* pada Tabel 4.3, hanya saja untuk pengkodeannya menggunakan "->" yang diganti pada baris 2 pada Kode Sumber 2.4.

```
1   IF (regexPattern3_1 = match) OR
    (regexPattern3_2 = match) THEN
2         set flagPostfix = 3
3         set postfix = ".O"
4         append sequence_code to AST
5         append count_level to level
6         append sourcecode_line, indeks_start,
          indeks_stop to line
7         append ctx.getText to sequence
    END IF
```

Kode Sumber 4.19 Fungsi `enterPostfixexpression` bagian 3

```
1   IF regexPattern11 = match THEN
2         IF (flagPostfix != 3 OR
          flagParametercall = 1) AND
          flagParameterpostfix = 0 THEN
3               flagPostfix = 1
4               postfix = "."
5         END IF
6   END IF
```

Kode Sumber 4.20 Fungsi `enterPostfixexpression` bagian 4

Kode Sumber 4.21 mengidentifikasi adanya penggunaan *decrement* dan *increment* pada kode program. Identifikasi dilakukan dengan melakukan pengecekan, apakah *postfix* tersebut mengandung **string** berupa "--" atau "++". Jika *postfix* tersebut mengandung **string** berupa "--" , maka diidentifikasikan sebagai sebuah *decrement*. Sebaliknya jika *postfix* tersebut mengadung string berupa "++" , maka diidentifikasikan sebagai sebuah *increment.* Variabel *tanda* pada Kode Sumber 4.21 adalah "++" atau "--" dan variabel *sequence_code* akan berisi sesuai dengan variabel *tanda.* Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.12 hingga Gambar C.16 untuk fungsi `enterPostfixexpression`.

Setelah mengunjungi AST dengan bantuan listener dan didapatkan kumpulan kode untuk satu sequence. Setelah itu

dilakukan pengecekan pada kelas ***praprocess*** untuk membedakan antara variabel biasa dan fungsi. Serta fungsi yang mempunyai parameter atau tidak.

```
1   IF postfixexpress contains tanda THEN
2         IF (flagSwitchcase > 0 AND
          flagCondition = 0) THEN
3               append sequence_code to AST
                append count_level to level
4               set postfix = NULL
5         END IF
6   END IF
```

Kode Sumber 4.21 Fungsi `enterPostfixexpression` bagian 5

4.2.2 Implementasi Metode Smith-Waterman

Metode Smith-Waterman digunakan untuk membandingkan dua buah *sequence* hasil olahan dari *listener* pada ANTLR. Tetapi metode ini mengidentifikasi kesamaan secara berurutan, sehingga jika ada peletakan kode yang dibalik kemungkinan tidak akan terdeteksi. Akan dilakukan modifikasi metode untuk mengatasi masalah tersebut.

```
FUNCTION Smithwaterman(sequence1, sequence2,
level1, level2)
 1   set variable match, mismatch, and w
 2   initialize matriks H
 3   FOR each row in matriks H
 4       FOR each col in matriks H
 5         IF secondCodeSequence equals
 6         firstCodeSequence AND
           secondLevel equals firstLevel
 7         THEN
 8            set max = matriks H + match
 9            set direction = 1
19         ELSE
11            set max= matriks H +mismatch
```

```
12              set direction = 9
13          END IF
14          IF max < (matriks H + w) THEN
15                set max = matriks H + w
16                set direction = 3
17          ELSE IF max < (matriks H + w) THEN
18                set max = matriks H + w
19                set direction = 2
20          END IF
21          set matriks H = max
22      END LOOP
23  END LOOP
24  jumlah = call function backtrack()
```

Kode Sumber 4.22 Metode Smith Waterman

```
FUNCTION Smith_modif(sequence1, sequence2,
level1, level2)
 1   set row = size of firstCodeSequence
 2   set coloumn = size of secondCodeSequence
 3   set w, match, mismatch
     initialize matriks H_modif, check_Col,
 4   check_Row
 5   FOR each row in matriks H_modif
 6      FOR each row in matriks H_modif
        IF  secondCodeSequence equals
 7      firstCodeSequence AND secondLeve equals
        firstLevel AND check_Col, check_Row
        equal 0 THEN
 8            set check_Col = -1
 9            set check_Row = -1
10            append row and col to index
11      END IF
12  call function backtrack_modif()
```

Kode Sumber 4.23 Metode Water-Smith Modifikasi

Kode Sumber 4.22 merupakan *pseudocode* metode *Smith-Waterman*. Variabel array dua dimensi *H* digunakan untuk menyimpan nilai hasil metode *smith-waterman*. Variabel *direction* digunakan untuk menyimpan arah hasil *insertion* (atas), *deletion* (kiri), atau kiri-atas. Pada baris 24, terdapat pemanggilan fungsi *backtracking* yang isinya dapat dilihat pada Kode Sumber 24. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.17.

Sedangkan Kode Sumber 4.23 merupakan *pseudocode* metode yang dimodifikasi. Pemodifikasiannya ditunjukkan pada baris 18 pada pengecekan kesamaan dua buah *sequence* dan ada penambahan kode pada baris 7-9. Variabel *check_Row* dan *check_Col* digunakan untuk menyimpan tanda bahwa sebuah kode telah dianggap sama. Indeks kode yang sama akan disimpan pada variabel *index*. Variabel itu digunakan untuk proses *backtracking*. Pada baris 12, terdapat pemanggilan fungsi *backtracking* modifikasi yang isinya dapat dilihat pada Kode Sumber 4.26. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.18.

4.2.3 Implementasi *Backtracking*

Backtracking dilakukan pada metode Smith-Waterman dan metode modifikasi. Pada Kode Sumber 4.24 yang merupakan *backtracking* Smith-Waterman menggunakan array satu dimensi dengan nama variabel *hasil* untuk menyimpan kode yang memiliki kesamaan, sedangkan *index_smith* digunakan untuk menyimpan indeksnya. Variabel *jumlah* adalah jumlah kode yang mirip, lalu digunakan untuk menghitung nilai kemiripan dengan membagi *jumlah* dengan jumlah *row* (panjang *source code* dosen) yang ditunjukkan pada Kode Sumber 4.25. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.19.

```
FUNCTION Int Backtrack()
 1    k = row and l = coloumn
 2    WHILE k >= 0 AND l >= 0
```

```
 3          IF direction = 2 THEN
 5                  decrease k
 7          ELSE IF direction= 3 THEN
10                  decrease l
11          ELSE
12            IF direction =  1 THEN
13                 append k,l to index_smith
14                 increase jumlah
15                 decrease k and l
17            END IF
18          END IF
19    END LOOP
20    return jumlah
```

Kode Sumber 4.24 Backtracking Smith-Waterman

```
FUNCTION double similarity()
1   get max(row,col)
2   set persentase  = 100.0 * jumlah / total
3   return persentase
```

Kode Sumber 4.25 Similarity Metode Smith-waterman

```
 1   clear array hs_i
 2   FOR v = 0 to maks
 3     FOR x = 0 to size of index
 4       set ind_i = indeks of firstCodeSequence
 5       set ind_j = indeks of secondCodeSequence
 6       set y = v
 7           IF ind_i = v THEN
 8               set flag = 1
 9               append ind_i to hs_i
10           END IF
11     END LOOP
12     IF flag = 0 THEN
```

```
13          append -1 to hs_i
14      END IF
15  END LOOP
16  set simi_modif = 100.0 * val / max(row,col)
```

Kode Sumber 4.26 Backtracking metode modifikasi

Untuk mempermudah pembacaan *output* kemiripan yang diurutkan sesuai indeks *sourcecode* dosen, maka pada variabel *index* harus diolah terlebih dahulu. Variabel *index* berisi indeks kode yang sama pada *source code* dosen maupun peserta didik. Pada Kode Sumber 4.26, dilakukan perulangan sebanyak panjang *source code* dosen untuk mencari pasangan indeks kode yang mirip antara *source code* dosen dan peserta didik. Hasil indeks tersebut akan dimasukkan ke dalam variabel *hs_i*. Sehingga indeks pada variabel *hs_i* mempunyai panjang yang sama dengan panjang *source code* dosen. Isi variabel *hs_i adalah* indeks pasangan yang mirip pada kode program dosen dan peserta didik. Jika *hs_i* ke-*v* berisi nilai -1, maka salah satu bagian dari kode program dosen tidak mirip dengan kode program peserta didik. Pada baris 16 dilakukan penghitungan nilai kemiripan dengan membagi nilai yang mirip dengan panjang *source code* paling panjang antara dosen dan mahasiswa. Detail kode sumber yang lebih lengkap dapat dilihat pada Lampiran Gambar C.20.

Pada saat penulisan hasil ke file teks (.txt), untuk hasil dari pengecekan dengan metode modifikasi, hasil kode kemiripan pada peserta didik dapat dilihat pada Kode Sumber 4.27 dan Kode Sumber 4.34 merupakan hasil kode kemiripan pada dosen. Variabel *listener*. S*equence* berisi kumpulan kode asli pada kode program peserta didik maupun dosen.

```
1   FOR indeks in matriks hs_i
2       IF matriks hs_i exists
3         IF sequence not equals "$" THEN
4            write "||-:-|" to file teks
5            write line + "code: "+ sequence
             to file teks
```

```
6           END IF
7         END IF
8      END LOOP
```

Kode Sumber 4.27 Penulisan kode yang mirip pada source code peserta didik berdasarkan hasil dari metode Smith-Waterman modifikasi

Hasil file teks ini yang akan dikirim ke sistem *E-Learning* yang kemudian akan digunakan sebagai data masukan pembuatan *feedback*. Sehingga dapat ditampilkan kode mana saja yang memiliki kemiripan. Data pada file teks yang digunakan adalah baris kode sumber, indeks mulai, indeks berhenti, dan kode isi *source code*.

4.2.4 Implementasi *Insert* dan *Select* Data pada Basis Data

Terdapat 3 tabel dalam basis data yaitu sc_dosen, SC_sequence_mhs, dan sc_similarity. Tabel sc_dosen digunakan untuk menyimpan *sequence* milik kode program dosen sedangkan SC_sequence_mhs kode program milik peserta didik. Sc_similarity digunakan untuk menyimpan nilai kemiripan kode program dosen dan peserta didik, kelas dan kode soal. Terdapat tiga kali proses *insert* yaitu *insert* tabel sc_dosen, SC_sequence_mhs, dan sc_similarity. Sedangkan *select* hanya pada tabel sc_dosen untuk dibandingkan dengan *sequence* peserta didik.

Kode Sumber 4.28 digunakan untuk memasukkan data kode program peserta didik ke dalam tabel SC_sequence_mhs. Hal sama seperti kode program dosen, hanya saja data akan dimasukkan ke dalam tabel SC_dosen. Sedangkan Kode Sumber 4.29 digunakan untuk memasukkan data kemiripan kode program peserta didik dan dosen ke dalam tabel SC_similarity.

Kode Sumber 4.30 digunakan untuk mengambil data kode program dosen. Data yang diambil adalah nama dan id file, sequence, level, dan baris kode program. Sequence dan level milik dosen akan dibandingkan dengan milik peserta didik, lalu dicari nilai kemiripannya.

```
1   set sessionFactory
2   set session of sessionFactory
3   beginTransaction of session
4   call entity SC_sequence_mhs
5   set attribute to entity
6   save attribute
7   commit transaction
8   close session
```

Kode Sumber 4.28 *Insert* data Kode Program Peserta Didik ke dalam SC_sequence_mhs.

```
1   set sessionFactory
2   set session of sessionFactory
3   beginTransaction of session
4   call entity SC_similarity
5   set attribute to entity
6   save attribute
7   commit transaction
8   close session
```

Kode Sumber 4.29 Insert Data Kemiripan dua buah kode program ke dalam SC_similarity.

```
1   set sessionFactory
2   set session of sessionFactory
3   beginTransaction of session
4   create query
5   setParameter to query
6   List results = query.list
7   Iterator it = results.iterator
8   get query result from entity SC_dosen
9   append query_result to array
10  commit transaction
11  close session
```

Kode Sumber 4.30 *Select* Kode Program Dosen dari tabel SC_dosen.

(Halaman ini sengaja dikosongkan)

BAB V
HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi pada tugas akhir ini. Hasil uji coba didapatkan dari implementasi pada bab 4 dengan skenario yang berbeda. Bab ini berisikan pembahasan mengenai lingkungan pengujian, data pengujian, dan uji kinerja.

5.1 Lingkungan Pengujian

Implementasi penghitungan kemiripan kode program dosen dan peserta didik menggunakan metode *Smith-Waterman dan* metode *Smith-Waterman* yang telah dimodifikasi menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Pengujian

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i7-3610QM CPU @ 2.30 GHz
	Memori	8 GB 1600 MHz DDR3
Perangkat Lunak	Sistem Operasi	Windows 8.1 Pro
	Perangkat Pengembang	Eclipse Luna , ANTLR V4, dan PostgreSQL V9.3

5.2 Data Pengujian

Subbab ini menjelaskan mengenai data yang digunakan pada uji coba. Seperti yang telah dijelaskan sebelumnya, data berasal dari tugas mata kuliah Pemrograman Berorientasi Objek (PBO) Kelas C Tahun Ajaran 2013/2014. Dataset yang digunakan adalah tiga jenis dataset, yaitu dataset class Invoice dan class

Account, serta dataset kode sumber yang tidak mirip. Jumlah dataset pertama adalah 31 buah kode program, sedangkan jumlah dataset kedua adalah 31 buah.

5.3 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini, perangkat akan diuji apakah sudah berjalan dengan benar dan bagaimana performa pada masing-masing skenario. Dan membandingkan skenario manakah yang memiliki hasil lebih baik. Terdapat 4 macam skenario uji coba, yaitu :

1. Pengujian pada setiap proses yang dikerjakan pada modul *student feedback system* apakah sudah berhasil atau tidak.
2. Penghitungan akurasi *similarity* antar kode program dataset jenis pertama menggunakan metode *Smith-waterman* dan Smith-Waterman modifikasi serta hasil kemiripan dua buah kode program
3. Penghitungan akurasi *similarity* antar kode program dataset jenis kedua menggunakan metode Smith-waterman dan Smith-Waterman modifikasi serta hasil kemiripan dua buah kode program
4. Penghitungan akurasi *similarity* antar kode program dataset jenis ketiga menggunakan metode Smith-waterman dan Smith-Waterman modifikasi serta hasil kemiripan dua buah kode program

5.3.1 Skenario Uji Coba 1

Skenario uji coba 1 adalah Pengujian pada setiap proses yang dikerjakan pada modul *student feedback system* apakah sudah berhasil atau tidak. Proses yang akan diuji ada 3, yaitu: praproses, metode Smith-Waterman, dan backtracking. Pada proses praproses dikatakan berhasil jika dapat menghasilkan sebuah output *sequence* yang merupakan kumpulan kode hasil *parsing* menggunakan ANTLR. Sedangkan metode Smith-Waterman dikatakan berhasil jika baik metode Smith-Waterman

dan Smith-Waterman yang dimodifikasi menghasilkan sebuah matriks yang akan digunakan untuk proses *backtracking* sehingga proses *backtracking* dapat berjalan. Sedangkan proses *backtracking* dikatakan berhasil jika dapat menghasilkan *output* nilai kemiripan antara kode sumber dosen dan mahasiswa. Selain itu, dapat menghasilkan sebuah *feedback* berupa file berekstensi (.txt) yang digunakan oleh *platform E-Learning* untuk menampilkan hasil kemiripan kode sumber dosen dan mahasiswa. Hasil pengujian dapat dilihat pada Tabel 5.2.

Tabel 5.2 Hasil Pengujian setiap Proses pada modul *Student Feedback System*

No	Proses	Keterangan
1	Praproses	Berhasil
2	Metode Smith-Waterman	Berhasil
	Metode Smith-Waterman yang dimodifikasi	Berhasil
3	Backtracking untuk Metode Smith-Waterman	Berhasil
	Backtracking untuk Metode Smith-Waterman yang dimodifikasi	Berhasil

5.3.2 Skenario Uji Coba 2

Skenario uji coba 2 adalah Penghitungan akurasi *similarity* antar kode menggunakan metode Smith-Waterman dan Smith-waterman modifikasi serta hasil kemiripan dua buah kode program. Dataset yang digunakan adalah dataset *class* Invoice. Dataset pertama ini memiliki keragaman yang lebih banyak dari dataset kedua.

Akurasi pada perhitungan kemiripan kode program dilakukan dengan menilai kemiripan dua buah kode program secara manual. Penilaian dilakukan dengan memberikan 2 jenis nilai *ground truth*, yaitu :

1. **0% - 45%** (kategori tidak mirip)
2. **46% - 100%** (kategori mirip)

Terdapat 31 kode program peserta didik dan 1 kode program dosen pada dataset jenis pertama yang memiliki total 31 nilai *similarity*. Dari masing-masing kode program peserta didik dicocokkan dengan kode program dosen baik melalui hasil *running* program tugas akhir atau secara manual. Lalu masing-masing kode program dipetakkan ke dalam 2 jenis kategori yang mengikuti nilai *ground truth*. Apabila nilai persentase kemiripan antar kedua kode program sesuai dengan kategori pada hasil penilaian manual, maka nilai tersebut dianggap benar (*True*). Sebaliknya apabila tidak sesuai, maka nilai tersebut dianggap salah (*False*).

Hasil penilaian secara manual dapat dilihat pada Tabel 5.3 dan penilaian akurasi kemiripan kode program pada uji coba 1 dapat dilihat pada Tabel 5.4.

Tabel 5.3 Penilian Kemiripan Dataset Pertama secara Manual

Nama_File	Self Assessment	
	Tidak Mirip <46%	Mirip >=46%
5112100002		v
5112100012		v
5112100013		v
5112100016		v
5112100026		v
5112100035		v
5112100038		v
5112100042	v	
5112100047		v
5112100056		v
5112100056		v

5112100057	v	
5112100059		v
5112100062		v
5112100068		v
5112100069		v
5112100073		v
5112100075		v
5112100080		v
5112100085	v	
5112100092		v
5112100097		v
5112100100		v
5112100115	v	
5112100120		v
5112100126		v
5112100127		v
5112100130		v
5112100138		v
5112100150		v
5112100166		v

Tabel 5.4 Penilaian Kemiripan Dataset Pertama menggunakan Program

Nama_File	Smith Waterman yang dimodif (%)	Smith Waterman (%)
5112100002	71.2	49.6
5112100012	71.2	45.6

5112100013	58.4	36.8
5112100016	67.2	44
5112100026	74.4	53.6
5112100035	72.8	49.6
5112100038	80	56.8
5112100042	25.6	20
5112100047	73.6	52
5112100056	80	56.8
5112100057	41.6	28
5112100059	80.8	60
5112100062	61.6	43.2
5112100068	77.6	56.8
5112100069	71.2	46.4
5112100073	73.6	52
5112100075	73.6	52
5112100080	100	100
5112100085	47.2	20
5112100092	60.8	49.6
5112100097	68.8	42.4
5112100100	71.2	45.6
5112100110	72	50.4
5112100115	32.8	22.4
5112100120	69.6	69.6
5112100126	74.4	52.8
5112100127	73.6	62.4
5112100130	60	49.6
5112100138	72.8	49.6

5112100150	67.2	50.4
5112100166	71.2	47.2

Tabel 5.5 Confusion Matriks Metode Smith-Waterman Uji Coba 2

		Prediksi	
		TM	M
Aktual	TM	4	0
	M	6	21

Sehingga dari Tabel 5.5 dapat dihitung nilai akurasi, presisi, dan recall.

$$\text{akurasi} = \frac{TP + TN}{total} = \frac{21 + 4}{31} = 80{,}64\%$$

$$\text{presisi} = \frac{TP}{\text{TP} + \text{FP}} = \frac{21}{21 + 0} = 100\%$$

$$\text{recall} = \frac{TP}{TP + FN} = \frac{21}{21 + 6} = 77{,}77\%$$

Tabel 5.6 Confusion Matriks Metode Water-Smith Modifikasi Uji Coba 2

		Prediksi	
		TM	M
Aktual	TM	3	1
	M	0	27

Sehingga dari Tabel 5.6 dapat dihitung nilai akurasi, presisi, dan recall.

$$\text{akurasi} = \frac{TP + TN}{total} = \frac{27 + 3}{31} = 96{,}7\%$$

$$\text{presisi } = \frac{TP}{\text{TP} + \text{FP}} = \frac{27}{27 + 1} = 96{,}4\%$$

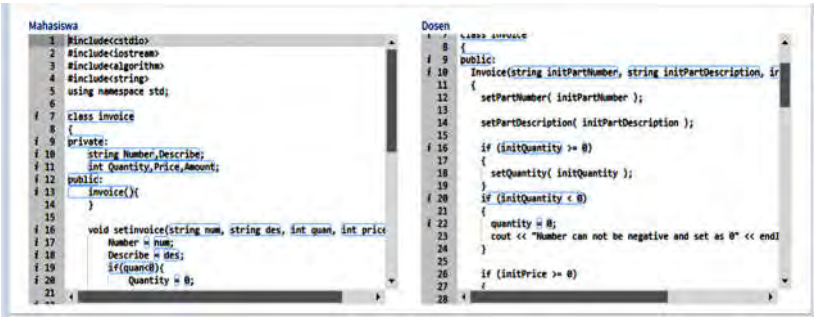
$$\text{recall } = \frac{TP}{TP + FN} = \frac{27}{27 + 0} = 100\%$$

Tabel 5.7 menunjukkan teks hasil kemiripan menggunakan metode yang sudah dimodifikasi yang sudah dipindahkan ke dalam tabel. Data keluaran menggunakan metode Smith-Waterman maupun metode yang sudah dimodifikasi mempunyai kode yang sama. "7 0 12 code: class invoice||-:-||" Angka pertama adalah baris line *source code* (angka 7), angka kedua adalah indeks kolom mulai pada *source code* (angka 0), angka ketiga adalah indeks kolom berhenti pada *source code* (angka 12), dan tulisan setelahnya "code:" adalah kode *sequence* ataupun hasil isi *source* code, sedangkan "||-:-||" sebagai pembeda antara kode kemiripan yang satu dengan lainnya. Pada kode sequence kata JV <nama variabel/literal> berarti berarti variabel tersebut berada pada kondisi, sedangkan =V atau =<nama variabel/literal> atau +=<nama variabel> berarti variabel tersebut merupakan variabel pada *assignment.*

Tabel 5.7 Hasil Teks Kemiripan Dua *Source code* Metode Smith-Waterman Modifikasi Uji Coba 1

Kode Sumber Mahasiswa
7 0 12 code: class invoice||-:-||12 0 5 code: public||-:-||16 17 26 code: Parameter : string num||-:-||16 29 38 code: Parameter : string des||-:-||16 41 48 code: Parameter : int quan||-:-||16 51 59 code: Parameter : int price||-:-||19 5 8 code: quan||-:-||19 10 10 code: 0||-:-||19 2 10 code: if(quan<0)||-:-||23 5 9 code: price||-:-||23 11 11 code: 0||-:-||20 12 12 code: =V ||-:-||20 14 14 code: =0||-:-||23 2 11 code: if(price<0)||-:-||22 16 16 code: =V ||-:-||24 11 11 code: =0||-:-||17 9 9 code: =V ||-:-||17 11 13 code: =num ||-:-||18 11 11 code: =V ||-:-||18 13 15 code: =des ||-:-||35 9 9 code: =V ||-:-||35 11 18 code: =Quantity ||-:-||13 0 9 code: int invoice()||-:-||35 20 24

code: =Price ||-:-||34 1 22 code: int getInvoiceAmount()||-:-||22 18 21 code: =quan ||-:-||24 9 9 code: =V ||-:-||26 13 13 code: =V ||-:-||9 0 6 code: private||-:-||10 4 16 code: string Number||-:-||10 11 25 code: string Describe||-:-||11 1 12 code: int Quantity||-:-||11 10 18 code: int Price||-:-||11 16 25 code: int Amount||-:-||40 0 9 code: int main()||-:-||42 1 9 code: return0
Kode Sumber Dosen
7 0 12 code: class Invoice||-:-||9 0 5 code: public||-:-||10 10 30 code: Parameter : string initPartNumber||-:-||10 33 58 code: Parameter : string initPartDescription||-:-||10 61 76 code: Parameter : int initQuantity||-:-||10 79 91 code: Parameter : int initPrice||-:-||16 8 19 code: initQuantity||-:-||16 24 24 code: 0||-:-||20 5 23 code: if(initQuantity<0)||-:-||20 8 19 code: initQuantity||-:-||20 23 23 code: 0||-:-||22 15 15 code: =V ||-:-||22 17 17 code: =0||-:-||30 5 20 code: if(initPrice<0)||-:-||32 12 12 code: =V ||-:-||32 14 14 code: =0||-:-||39 15 15 code: =V ||-:-||39 17 30 code: =initPartNumber ||-:-||49 20 20 code: =V ||-:-||49 22 40 code: =initPartDescription ||-:-||59 13 13 code: =V ||-:-||59 15 26 code: =initQuantity ||-:-||62 2 18 code: int getQuantity()||-:-||69 10 10 code: =V ||-:-||72 2 15 code: int getPrice()||-:-||85 19 19 code: =V ||-:-||85 21 29 code: =iQuantity ||-:-||85 33 38 code: =iPrice ||-:-||90 0 6 code: private||-:-||91 2 18 code: string partNumber||-:-||92 2 23 code: string partDescription||-:-||93 2 13 code: int quantity||-:-||94 2 10 code: int price||-:-||95 2 18 code: int invoiceAmount||-:-||98 0 9 code: int main()||-:-||100 1 9 code: return0 72



Gambar 5.1 UI Feedback Berupa Tanda pada Text yang Mirip pada NRP 5112100085 Uji Coba 2

Isi Tabel 5.7 merupakan hasil milik *source code* "5112100085.cpp". Menurut pengamatan manual *source code ini* termasuk dalam jenis tidak mirip sedangkan pada hasil programnya terdeteksi mirip. Hal ini dikarenakan karena pada hasil program, terdapat pendeteksian terhadap variabel (*unqualifiedid*) sehingga menambah nilai kemiripan. Sedangkan Gambar 5.1 merupakan UI *feedback* berupa tanda pada text yang mirip pada NRP 5112100085.

5.3.3 Skenario Uji Coba 3

Skenario uji coba 3 adalah penghitungan akurasi kemiripan antara kode program dosen dan peserta didik menggunakan metode Smith-Waterman dan Smith-Waterman modifikasi serta hasil kemiripan dua buah kode program. Dataset yang digunakan adalah dataset *Class* Account. Sebagian besar dataset ini mempunyai *source code* yang mempunyai kemiripan besar.

Akurasi pada perhitungan kemiripan kode program dilakukan dengan menilai kemiripan dua buah kode program secara manual. Penilaian dilakukan dengan memberikan 2 jenis nilai *ground truth*, yaitu :

1. **0% - 45%** (kategori tidak mirip)
2. **46% - 100%** (kategori mirip)

Terdapat 31 kode program peserta didik dan 1 kode program dosen pada dataset jenis pertama yang memiliki total 31 nilai *similarity*. Dari masing-masing kode program peserta didik dicocokkan dengan kode program dosen baik melalui hasil *running* program tugas akhir atau secara manual. Lalu masing-masing kode program dipetakkan ke dalam 2 jenis kategori yang mengikuti nilai *ground truth*. Apabila nilai persentase kemiripan antar kedua kode program sesuai dengan kategori pada hasil penilaian manual, maka nilai tersebut dianggap benar (*True*). Sebaliknya apabila tidak sesuai, maka nilai tersebut dianggap salah (*False*).

Hasil penilaian secara manual dapat dilihat Tabel 5.8 dan penilaian akurasi kemiripan kode program pada uji coba 3 dapat dilihat Tabel 5.9.

Tabel 5.8 Penilaian Kemiripan Dataset Kedua secara Manual

Nama_File	Self Assesment	
	Tidak Mirip <46%	Mirip >=46%
5112100002		v
5112100012		v
5112100013		v
5112100016		v
5112100026		v
5112100035		v
5112100038		v
5112100042		v
5112100047		v
5112100056		v
5112100056		v

5112100057		v
5112100059		v
5112100062		v
5112100068		v
5112100069		v
5112100073		v
5112100075		v
5112100080		v
5112100085		v
5112100092	v	
5112100097		v
5112100100		v
5112100115	v	
5112100120		v
5112100126		v
5112100127		v
5112100130		v
5112100138		v
5112100150		v
5112100166		v

Tabel 5.9 Penilian Kemiripan Dataset Kedua menggunakan Program

Nama_File	Smith Waterman (%)	Smith Waterman Modif (%)
5112100002	77,58	81,03

5112100012	75,86	79,31
5112100013	74,13	81,03
5112100016	72,41	72,41
5112100026	77,58	77,58
5112100035	77,58	77,58
5112100038	72,41	77,58
5112100042	72,41	74,13
5112100047	72,41	74,13
5112100056	67,24	72,41
5112100057	50,84	67,79
5112100059	75,86	75,86
5112100062	55,73	57,37
5112100068	81,03	94,82
5112100069	67,24	86,20
5112100073	67,24	86,20
5112100075	67,24	86,20
5112100080	96,55	96,55
5112100085	58,62	67,24
5112100092	24,56	42,10
5112100097	75,86	79,31
5112100100	70,68	77,58
5112100110	81,03	77,58
5112100115	2,45	24,53
5112100120	65,62	81,25
5112100126	74,13	77,58
5112100127	59,42	71,01
5112100130	65,51	75,86

5112100138	59,74	63,63
5112100150	70	70
5112100166	81,03	93,10

Tabel 5.9 menunjukkan hasil uji coba penghitungan kemiripan dua buah *source code* menggunakan dua metode yaitu Smith-Waterman dan Smith-Waterman Modifikasi. Pada metode Smith-Waterman memiliki nilai kemiripan lebih kecil karena metode ini hanya membandingkan secara *sequence* atau urut. Jadi tidak dapat mendeteksi kode yang peletakannya dibolak-balik.

Tabel 5.10 Confusion Matriks Metode Water-Smith Uji Coba 3

		Prediksi	
		TM	M
Aktual	TM	2	0
	M	0	29

Sehingga dari Tabel 5.10 dapat dihitung nilai akurasi, presisi, dan recall.

$$\text{akurasi} = \frac{TP + TN}{total} = \frac{29 + 2}{31} = 100\%$$

$$\text{presisi} = \frac{TP}{\text{TP} + \text{FP}} = \frac{29}{29 + 0} = 100\%$$

$$\text{recall} = \frac{TP}{TP + FN} = \frac{29}{29 + 0} = 100\%$$

Tabel 5.11 Confusion Matriks Metode Water-Smith Modifikasi Uji Coba 3

		Prediksi	
		TM	M
Actual	TM	2	0
	M	0	29

Sehingga dari Tabel 5.11 dapat dihitung nilai akurasi, presisi, dan recall.

$$\text{akurasi} = \frac{TP + TN}{total} = \frac{31 + 0}{31} = 100\%$$

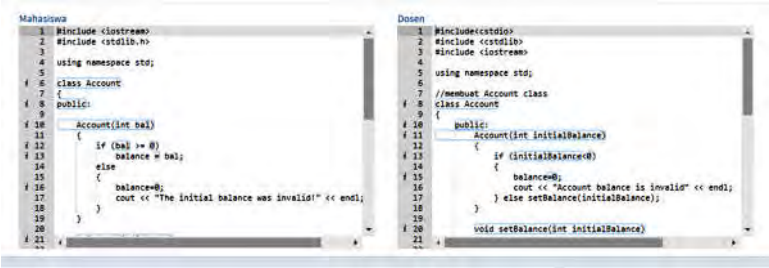
$$\text{presisi} = \frac{TP}{\text{TP} + \text{FP}} = \frac{31}{31 + 0} = 100\%$$

$$\text{recall} = \frac{TP}{TP + FN} = \frac{31}{31 + 0} = 100\%$$

Gambar 5.2 menunjukkan UI feedback berupa tanda pada text yang mirip pada NRP 5112100002 Uji Coba 2. Tabel 5.12 menunjukkan teks hasil kemiripan menggunakan metode yang sudah dimodifikasi yang sudah dipindahkan ke dalam tabel. Data keluaran menggunakan metode Smith-Waterman maupun metode yang sudah dimodifikasi mempunyai kode yang sama. "6 0 12 code: class Account||-:-||" Angka pertama adalah baris line *source code* (angka 6), angka kedua adalah indeks kolom mulai pada *source code* (angka 0), angka ketiga adalah indeks kolom berhenti pada *source code* (angka 12), dan tulisan setelahnya "code:" adalah kode *sequence* ataupun hasil isi *source* code, sedangkan "||-:-||" sebagai pembeda antara kode kemiripan yang satu dengan lainnya. Pada kode sequence kata JV <nama variabel/literal> berarti berarti variabel tersebut berada pada kondisi, sedangkan =V atau =<nama variabel/literal> atau +=<nama variabel> berarti variabel tersebut merupakan variabel pada *assignment.*

Tabel 5.12 Hasil Teks Kemiripan Dua *Source code* Metode Smith-Waterman Modifikasi Uji Coba 3

Kode Sumber Mahasiswa
6 0 12 code: class Account||-:-||8 0 5 code: public||-:-||10 0 16 code: Account Account(intbal)||-:-||10 9 15 code: Parameter : int bal||-:-||12 6 8 code: bal||-:-||12 13 13 code: 0||-:-||13 11 11 code: =V ||-:-||16 11 11 code: =0||-:-||21 1 20 code: void credit(intbal)||-:-||21 13 19 code: Parameter : int bal||-:-||26 12 18 code: Parameter : int bal||-:-||28 5 7 code: bal||-:-||28 12 18 code: balance||-:-||29 10 11 code: -=V ||-:-||29 12 14 code: -=bal ||-:-||23 9 10 code: +=V ||-:-||23 11 13 code: +=bal ||-:-||34 1 16 code: int getBalance()||-:-||36 2 16 code: returnbalance||-:-||39 0 6 code: private||-:-||40 1 11 code: int balance||-:-||43 0 9 code: int main()||-:-||44 1 9 code: return0
Kode Sumber Dosen
8 0 12 code: class Account||-:-||10 1 6 code: public||-:-||11 0 28 code: Account Account(intinitialBalance)||-:-||11 10 27 code: Parameter : int initialBalance||-:-||13 12 25 code: initialBalance||-:-||13 27 27 code: 0||-:-||15 15 15 code: =V ||-:-||15 16 16 code: =0||-:-||20 2 36 code: void setBalance(intinitialBalance)||-:-||20 18 35 code: Parameter : int initialBalance||-:-||25 12 26 code: Parameter : int debitAmount||-:-||27 9 19 code: debitAmount||-:-||27 21 27 code: balance||-:-||28 17 18 code: -=V ||-:-||28 19 29 code: -=debitAmount ||-:-||33 12 13 code: +=V ||-:-||33 14 25 code: +=creditAmount ||-:-||36 2 17 code: int getBalance()||-:-||38 3 17 code: returnbalance||-:-||41 1 7 code: private||-:-||42 4 14 code: int balance||-:-||45 0 9 code: int main()||-:-||47 0 8 code: return0



Gambar 5.2 UI Feedback Berupa Tanda pada Text yang Mirip pada NRP 5112100002 Uji Coba 2

5.3.4 Skenario Uji Coba 4

Skenario uji coba 4 adalah penghitungan akurasi kemiripan antara kode program dosen dan peserta didik menggunakan metode Smith-Waterman dan Smith-Waterman modifikasi serta hasil kemiripan dua buah kode program. Dataset yang digunakan adalah kumpulan kode sumber yang tidak mirip yang dibandingkan dengan kode sumber dosen *class* Invoice.

Akurasi pada perhitungan kemiripan kode program dilakukan dengan menilai kemiripan dua buah kode program secara manual. Penilaian dilakukan dengan memberikan 2 jenis nilai *ground truth*, yaitu :

3. **0% - 45%** (kategori tidak mirip)
4. **46% - 100%** (kategori mirip)

Terdapat 9 kode program peserta didik dan 1 kode program dosen pada dataset jenis pertama yang memiliki total 9 nilai *similarity*. Dari masing-masing kode program peserta didik dicocokkan dengan kode program dosen baik melalui hasil *running* program tugas akhir atau secara manual. Lalu masing-masing kode program dipetakkan ke dalam 2 jenis kategori yang mengikuti nilai *ground truth*. Apabila nilai persentase kemiripan antar kedua kode program sesuai dengan kategori pada hasil penilaian manual, maka nilai tersebut dianggap benar (*True*). Sebaliknya apabila tidak sesuai, maka nilai tersebut dianggap salah (*False*).

Hasil penilaian secara manual dapat dilihat Tabel 5.12 dan penilaian akurasi kemiripan kode program pada uji coba 3 dapat dilihat Tabel 5.13.

Tabel 5.13 Penilaian Kemiripan Dataset Ketiga secara Manual

Nama_File	Self Assesment	
	Tidak Mirip <46%	Mirip >=46%
5112100001	v	
5112100002	v	
5112100003	v	
5112100004	v	
5112100012	v	
5112100059	v	
5112100110	v	
5112100166	v	
5112100168	v	

Tabel 5.14 Penilian Kemiripan menggunakan Program pada Dataset Ketiga dengan Kode Sumber yang tidak mirip sama sekali

Nama_File	Smith Waterman (%)	Smith Waterman Modif (%)
5112100001	0	0
5112100002	4,23	5,08
5112100004	6,77	16,10
5112100110	5,08	16,10

Tabel 5.15 Penilian Kemiripan menggunakan Program pada Dataset Ketiga dengan Kode Sumber Kelas Invoice yang tidak mirip

Nama_File	Smith Waterman (%)	Smith Waterman Modif (%)
5112100012	8,47	17,79
5112100168	42,37	42,37

Tabel 5.16 Penilian Kemiripan menggunakan Program pada Dataset Ketiga dengan Kode Sumber Kelas Account yang dibandingkan dengan Kelas Invoice

Nama_File	Smith Waterman (%)	Smith Waterman Modif (%)
5112100003	36,44	41,52
5112100059	22,88	39,83
5112100166	22,88	40,67

Tabel 5.14 menunjukkan nilai kemiripan pada kode sumber yang tidak mirip sama sekali. Kode sumber 5112100001 memiliki nilai kemiripan 0% baik menggunakan metode Smith-Waterman dan Smith-Waterman yang dimodifikasi dikarenakan memiliki isi yang kosong. Sedangkan tabel 5.15 menunjukkan nilai kemiripan kode sumber *Class* Invoice yang tidak mirip karena tidak adanya fungsi-fungsi sesuai soal Invoice pada kode sumber 51121000059, sedangkan kode sumber 5112100168 memiliki fungsi yang kurang lengkap dan tipe variabel yang berbeda. Lebih jelasnya kode sumber dapat dilihat pada halaman L Ada beberapa kode sumber *Class* Account yang dibandingkan dengan kode sumber dosen *Class* Invoice yaitu Kode Sumber dengan NRP 5112100003, 5112100059, 5112100166. Nilai

kemiripannya dapat dilihat pada Tabel 5.16. Kode sumber dataset 3 dapat dilihat pada Lampiran B. Data Uji.

Tabel 5.17 Confusion Matriks Metode Water-Smith Uji Coba 3

		Prediksi	
		TM	M
Actual	TM	9	0
	M	0	0

Sehingga dari Tabel 5.14 dapat dihitung nilai akurasi, presisi, dan recall.

$$\text{akurasi} = \frac{TP + TN}{total} = \frac{9 + 0}{9} = 100\%$$

$$\text{presisi} = \frac{TP}{\text{TP} + \text{FP}} = \frac{9}{9 + 0} = 100\%$$

$$\text{recall} = \frac{TP}{TP + FN} = \frac{9}{9 + 0} = 100\%$$

Tabel 5.18 Confusion Matriks Metode Water-Smith yang dimodifikasi Uji Coba 3

		Prediksi	
		TM	M
Actual	TM	9	0
	M	0	0

Sehingga dari Tabel 5.15 dapat dihitung nilai akurasi, presisi, dan recall.

$$\text{akurasi} = \frac{TP + TN}{total} = \frac{9 + 0}{9} = 100\%$$

$$\text{presisi} = \frac{TP}{\text{TP} + \text{FP}} = \frac{9}{9 + 0} = 100\%$$

$$\text{recall} = \frac{TP}{TP + FN} = \frac{9}{9 + 0} = 100\%$$

5.4 Evaluasi Umum Skenario Uji Coba

Berdasarkan 4 skenario uji coba yang telah dilakukan, dapat diketahui bahwa tingkat ketepatan sistem untuk menilai kemiripan dua buah kode program mahasiswa sudah sangat baik yaitu:

1. Pada uji coba 2, 3, dan 4 pendeteksian menggunakan metode Smith-Watermen memiliki nilai kemiripan yang lebih kecil karena tidak dapat mengidentifikasi peletakkan kode program yang dibolak-balik.
2. Pada uji coba 2, pendeteksian menggunakan metode yang dimodifikasi terdapat satu *source code* yang dianggap mirip padahal dalam pengecekan secara manual dianggap tidak mirip. Hal tersebut dikarenakan kesamaan kode program ditemukan dalam pendeklarasian variabel.
3. Kode sumber yang memiliki panjang yang besar, mempunyai nilai kemiripan yang lebih kecil.
4. Nilai akurasi sistem dapat dihitung dengan menggunakan rumus rata-rata sebuah nilai yang ditunjukkan pada Persamaan 5.1. Sehingga didapatkan nilai akurasi untuk metode Smith-Waterman adalah 93,54%, sedangkan untuk metode Smith-Waterman yang dimodifikasi adalah 98,9%.

.

$$\overline{akurasi} = \frac{\sum_{i=2}^{4} akurasi\_ujicoba(i)}{3} \qquad (5.1)$$

Hal yang ditemukan selama pengujian:

1. Ada modifikasi Smith-Waterman untuk meningkatkan akurasi deteksi kemiripan kode program yang urutannya dibolak-balik.
2. Pada uji coba 2, terdapat *source code* yang menggunakan *ternary operator* pada *if-else*.

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak nantinya.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba adalah sebagai berikut:

1. Sistem *back-end E-Learning* pada modul *Student Feedback System* dapat menilai kemiripan dua buah *source code* C++ berdasarkan strukturnya.
2. Sistem *back-end E-Learning* pada modul *Student Feedback System* dapat memberikan *feedback* berupa bagian mana pada dua buah kode sumber yang mirip dengan cara *backtracking* untuk metode metode Smith-Waterman dan mengambil indeks pada *list* kode yang mirip untuk metode Smith-Waterman modifikasi.
3. Setiap proses dalam Sistem *back-end E-Learning* pada modul *Student Feedback System* dapat dinyatakan berhasil karena menghasilkan output yang sesuai.
4. Metode Smith-Waterman yang telah dimodifikasi terbukti lebih baik untuk mendeteksi kemiripan dua buah *source code C++* karena mempunyai nilai akurasi yang lebih bagus.
5. Nilai akurasi sistem *back-end E-Learning* pada modul *Student Feedback System* adalah untuk metode Smith-Waterman adalah 93,54%, sedangkan untuk metode Smith-Waterman yang dimodifikasi adalah 98,9%.

6.2 Saran

Saran yang diberikan terkait pengembangan pada tugas akhir ini adalah:

1. Seharusnya bisa mendeteksi penggunaan ***if-else*** dengan *ternary operator*.
2. Seharusnya pemberian level pada isi ***if*** dan ***else if*** adalah sama, tetapi pada *grammar C++* terdapat penambahan level sebanyak 1 setiap isi ***else if*** **.**
3. ***Grammar C++*** yang digunakan masih terlalu umum, karena banyak *rule* yang mempunyai isi yang sama.
4. Dataset yang digunakan masih kurang variatif, sehingga banyak fungsi-fungsi pada listener tidak digunakan.

DAFTAR PUSTAKA

[1] L. Jiang, G. Mishergi dan Z. Su, "DECKARD: Scalable and Accurate Tree-based Detection of Code Clones," dalam *ICSE '07 Proceeding of 29th International Conference on Software Engineering*, 2007.

[2] L. P. Zhang dan D. S. Liu, "AST-based Multi-language Plagiarism Detection Method," dalam *Software Engineering and Service Science (ICSESS), 2013 4th IEEE International Conference*, Beijing, 2013.

[3] O. W. Purbo, Teknologi E-learning Berbasis PHP dan MySQL, Jakarta: Elex Media Komputindo, 2002.

[4] T. Paar, "ANTLR," [Online]. Available: http://www.antlr.org. [Diakses 29 Januari 2016].

[5] T. Parr, "Parse-Tree Listeners and Visitor," dalam *The Definitive ANTLR 4 Reference*, 2013, pp. 17-20.

[6] "The PostgreSQL Global Development Group," 2006. [Online]. Available: http://www.postgresql.org/docs/current/static/intro-whatis.html. [Diakses 21 Desember 2015].

[7] E. Ayguade, J. J. Navarro dan D. Jimenez-Gonzales, "DAC,' [Online]. Available http://docencia.ac.upc.edu/master/AMPP/slides/ampp_sw_presentation.pdf. [Diakses 11 Maret 2016].

[8] "Github Grammar ANTLR," [Online]. Available: https://github.com/antlr/grammars-v4. [Diakses 29 Desember 2015].

LAMPIRAN A. ATRIBUT ENTITAS BASIS DATA

Tabel A. 1 Atribut Entitas SC_sequence_mhs

No	Atribut	Penjelasan
1	id_seq_mhs	ID tiap kode program peserta didik yang digenerate melalui UUID
2	extension_mhs	Ekstensi dari file kode program peserta didik (umumnya .cpp)
3	file_name_mhs	Nama file kode program peserta didik
4	hashcode_mhs	Nilai hashcode tiap file kode program peserta didik (bersifat unik)
5	kelas_mhs	Kelas dimana mata kuliah diambil oleh peserta didik
6	last_modified_mhs	*Timestamp,* waktu dimana file kode program peserta didik terakhir dibuat
7	nrp_mhs	NRP peserta didik
8	path_mhs	Letak file kode program peserta didik disimpan
9	size_mhs	Ukuran file kode program peserta didik (*byte*)
10	sequence_mhs	String panjang hasil konversi *sequence*
11	level_mhs	String panjang hasil perhitungan level tiap elemen pada *sequence*
12	line_mhs	String baris kode sumber peserta didik
13	kode_soal	String Kode Soal

Tabel A. 2 Atribut Entitas SC_dosen

No	Atribut	Penjelasan
1	id_seq_dosen	ID tiap kode program dosen yang digenerate melalui UUID
2	extension	Ekstensi dari file kode program dosen (umumnya .cpp)
3	file_name	Nama file kode program dosen
4	hashcode	Nilai hashcode tiap file kode program dosen (bersifat unik)
5	kodesoal	String kode soal
6	last_modified	*Timestamp,* waktu dimana file kode program dosen terakhir dibuat
7	path	Letak file kode program dosen disimpan
8	size	Ukuran file kode program dosen (*byte*)
9	sequence	String panjang hasil konversi *sequence*
10	level	String panjang hasil perhitungan level tiap elemen pada *sequence*
11	line	String baris kode sumber dosen

Tabel A. 3 Atribut Entitas SC_similarity

No	Atribut	Penjelasan
1	id_similarity	ID nilai *similarity* tiap kode program mahasiswa yang digenerate melalui UUID
2	file_dosen	nama file kode program dosen
3	nrp2	nrp peserta didik

4	similarity_value_smith	Nilai presentase similarity dua buah kode program dengan metode Smith-Waterman
5	similarity_value_modif	Nilai presentase similarity dua buah kode program dengan metode Smith-Waterman modifikasi
6	kelas	String kelas mahasiswa
7	kodesoal	String kode soal

Halaman ini sengaja dikosongkan)

LAMPIRAN B. DATA UJI

Tabel B. 1 Kode Sumber 5112100001 pada Dataset Ketiga

```
#include<stdio.h>
```

Tabel B. 2 Kode Sumber 5112100002 pada Dataset Ketiga

```
#include <iostream>
#include <string>

using namespace std;

int main(){

}
```

Tabel B. 3 Kode Sumber 5112100003 pada Dataset Ketiga

```
#include<cstdio>
#include <cstdlib>
#include <iostream>

using namespace std;

//membuat Account class
class Account
{
      public:
            Account(int initialBalance)
            {
              if (initialBalance<0)
              {
                  balance=0;
                  cout << "Account balance is
                  invalid" << endl;
               } else
                  setBalance(initialBalance);
            }
```

```
            void setBalance(int initialBalance)
            {
                   balance=initialBalance;
            }

            void debit(int debitAmount)
            {
                  if (debitAmount>balance)
                        cout<<"Your balance is
                        not enough.."<<endl;
                  else balance-=debitAmount;
            }

            void credit(int creditAmount)
            {
                  balance+=creditAmount;
            }

            int getBalance()
            {
                  return balance;
            }

      private:
            int balance;
};

int main()
{
return 0;
}
```

Tabel B. 4 Kode Sumber 5112100004 pada Dataset Ketiga

```
#include <iostream>

using namespace std;

void nama()
```

```
{
	cout << "nama" << endl;
}

void tambah(int a, int b)
{
	cout << a + b << endl;
}
int main()
{
	int a = 10;

	int bil1, bil2;
	int sisa;
	cout <<"Masukkan bil 1	: ";
	cin >> bil1;

	//if else
	if (bil1 <= 10 || bil1 >= 6)
	{
		char k;
		bil2 = bil1;

	}
	else if(bil1 > 0 && bil1 < 5)
	{
		bil2 = 2;
		char k;
	}
	else
	{
		int rahma;
		bil1 += 4;
	}

	//do while
```

```
      int k = 0;
      char z;
      do
      {
            cout << "do-while ";
            cout << k <<"\n";
            k++;

      }while (k <bil2);

      //for
}
}
```

Tabel B. 5 Kode Sumber 5112100012 pada Dataset Ketiga

```
#include <iostream>
#include <string>
using namespace std;

class Invoice
{
      private :
            string partNumber;
            string partDesc;
            int quantity;
            int price;

      public :
            Invoice ();
            Invoice(string prtNm,int qnty,
            string prtDs, int prc) {
            }

};
int main (){
}
```

Tabel B. 6 Kode Sumber 5112100059 pada Dataset Ketiga

```
#include<iostream>
using namespace std;
class Account {
      public:
            Account (int saldo) {
                  if (saldo >= 0) {
                        balance = saldo;
                  }
                  else {
                        balance = 0;
                        cout << "Error! Uang
                        Anda sedikit" << endl;
                  }
            }
            void credit (int duit) {
                  balance = balance + duit;
            }
            void debit (int duit) {
                  if (duit <= balance) {
                        balance = balance -
                        duit;
                  }
                  else cout << "Uang Anda
                  tidak mencukupi" << endl;
            }

            int getBalance() {
                  return balance;
            }
      private: //biar ga bisa diubah seenaknya
            int balance;
};

int main(){
}
```

Tabel B. 7 Kode Sumber 5112100110 pada Dataset Ketiga

```
#include "iostream"

using namespace std;

class contoh {
      static int Y;
      int X;
      public:
            void setcontoh(int input)
            {
                  X = input;
                  cout << X;
            }
      private:
            int z;

};

int main()
{

}
```

Tabel B. 8 Kode Sumber 5112100166 pada Dataset Ketiga

```
#include <iostream>

using namespace std;

class Account
{
      public:
      Account (int blc)
            {
                  if (blc >= 0)
```

```
                  setBalance(blc);
            else
                  {
                  setBalance(0);
            cout << "Initial value cannot be
            less than 0" << endl;
            }
      }
      void credit(int blc)
      {
                  balance += blc;
      }
      void debit(int blc)
      {
            if (blc > balance)
                  cout << "Debit amount
                  exceeded account balance"
                  << endl;
            else
                  balance -= blc;
      }
      int getBalance()
      {
            return balance;
      }
      int setBalance(int blc)
      {
            balance = blc;
      }
      private:
            int balance;
};

int main() {
  return 0;
}
```

Tabel B. 9 Kode Sumber 5112100168 pada Dataset Ketiga

```
#include <cstdio>
#include <cstdlib>
#include <iostream>

using namespace std;

class Invoice
{
public:
  Invoice()
  {
    setPartNumber( initPartNumber );

    if (initQuantity >= 0)
    {
      setQuantity( initQuantity );
    }
    if (initQuantity < 0)
    {
      quantity = 0;
      cout << "Number can not be negative
       and set as 0" << endl;
    }
  }

  void setPartNumber(string initPartNumber)
  {
    partNumber = initPartNumber;
  }

  string getPartNumber()
  {
    return partNumber;
  }
```

```
  void setPrice( double initPrice )
  {
    price = 1000.0;
  }

  double getPrice()
  {
    return 1.0;
  }


private:
  char partNumber;
  char partDescription;
  double quantity;
  double price;
  double invoiceAmount;
};

int main()
{
     return 0;
}
```

Tabel B. 10 Kode Sumber jawaban dosen pada Dataset Ketiga

```
#include <cstdio>
#include <cstdlib>
#include <iostream>

using namespace std;

class Invoice
{
public:
 Invoice(string initPartNumber, string
 initPartDescription, int initQuantity, int
```

```
 initPrice)
 {
    setPartNumber( initPartNumber );

    setPartDescription( initPartDescription
);

    if (initQuantity >= 0)
    {
      setQuantity( initQuantity );
    }
    if (initQuantity < 0)
    {
      quantity = 0;
      cout << "Number can not be negative
       and set as 0" << endl;
    }

    if (initPrice >= 0)
    {
      setPrice( initPrice );
    }
    if (initPrice < 0)
    {
      price = 0;
      cout << "Price can not be negative and
       set as 0" << endl;
    }
  }
  void setPartNumber(string initPartNumber)
  {
    partNumber = initPartNumber;
  }

  string getPartNumber()
  {
```

```
    return partNumber;
  }
  void setPartDescription (string
  initPartDescription)
  {
    partDescription = initPartDescription;
  }

  string getPartDescription()
  {
    return partDescription;
  }

  int getQuantity()
  {
    return quantity;
  }

  void setPrice( int initPrice )
  {
    price = initPrice;
  }

  int getPrice()
  {
    return price;
  }
  int getInvoiceAmount(int iQuantity, int
  iPrice)
  {
    if((iQuantity<0)||(iPrice<0))
    {
      cout << "Incorrect args!" << endl;
    }
    else
    {
```

```
      invoiceAmount = iQuantity * iPrice;
    }
    return invoiceAmount;
  }

private:
  string partNumber;
  string partDescription;
  int quantity;
  int price;
  int invoiceAmount;
};

int main()
{
      return 0;
}
```

LAMPIRAN C. KODE PROGRAM

```
enterDeclaration(ctx)
 1   count_level = 0
 2   ast.append("A ( ")
 3   level.append(count_level + " " + count_level
     + " ")
 4   line.append(ctx.getStart().getLine() + " " +
     "-1 ")
 5   sequence.add("$")
 6   sequence.add("$")
 7   count_B = 0

exitDeclaration(ctx)
  1  if count_L > 0
  2          ast.append (") ")
  3          line.append(") ")
  4          count_level --
  5          level.append(count_level + " ")
  6          sequence.add("$")
  7          count_L = 0
  8  ast.append(")  ")
  9  count_level--
 10  level.append(count_level + " ")
 11  line.append("-1 ")
```

Gambar C. 1 Kode Sumber Fungsi `enterDeclaration` dan `exitDeclaration` pada listener

```
enterClasskey(ctx)
 1   classkey = ctx.getText()
```

```
enterClassname(ctx)
 1   if ctx.getText().contains("string")
 2         classname = 0
 3   else
 4         ast.append("NK ")
 5         level.append(count_level + " "
           + count_level + " ")
           line.append(spl_line(ctx.getText()+"
 6         "+ " ")
           sequence.add(classkey + " " +
 7         ctx.getText())

enterClassspecifier(ctx)
 1   flag_class = 1
 2   ast.append("K ( ")
     level.append(count_level + " " +
 3   count_level + " ")
 4   line.append("-1 " + "-1 ")
 5   sequence.add("$")
 6   sequence.add("$")
 7   count_level++

exitClassspecifier(ctx)
 1   flag_class = 0
 2   ast.append(")  ")
 3   count_level--
 4   level.append(count_level + " ")
 5   line.append("-1 ")
 6   sequence.add("$")
```

Gambar C. 2 Kode Sumber Fungsi classname, classkey, classpecifier pada listener

```
enterAccesspecifier(ctx)
 1    if countAccessspec = 0
        line_sourcecode =
 2      spl_lineSC(ctx.getStart())
 3      seq = " L: " + ctx.getText()  + " ( "
 4      ast.append(seq)
 5      level.append(level_count)
 6      level_count++
 7      line.append(line_sourcecode)
 8      countAccessspec = 1
 9      sequence.add(ctx.getText())
10      sequence.add("$")
11    else if countAccessspec > 0
        line_sourcecode = "-1 " +
12      spl_lineSC(ctx.getStart())
13      seq =  ") L: " + ctx.getText()  + " ( "
14      ast.append(seq)
15      level_count--
16      level.append(level_count)
17      level_count++
18      level.append(level_count)
19      line.append(line_sourcecode)
20      sequence.add("$")
21      countAccessspec = 0
22      sequence.add(ctx.getText())
23      sequence.add("$")
```

Gambar C. 3 Kode Sumber `enterAccesspecifier` pada listener

```
enterDeclarator(ctx)
1   tipe = st.peek() ;
    if ctx.getText().contains("(")  &&
2   ctx.getText().contains(")")
          if ctx.getText().contains(classname
3         + "()") && flag_class!=0)
4               if ctx.getText().contains("~"))
5                     ast.append("B:Dest ")
6                else
7                      ast.append("B:Const ")
8                sequence.add(ctx.getText())
9          else
                 ast.append("B:" + tipe +
10                "() ")
                 sequence.add(tipe + " " +
11                ctx.getText())
12  else
13         if count_D > 0

                  sequence.add("Parameter :
14                " + tipe + " " +
                  ctx.getText())
15         else
                 sequence.add(tipe + " " +
16                ctx.getText())
17         ast.append("B:" + tipe + " ")
18  level.append(count_level +" ")
19  line.append(ctx.getStart().getLine()+ " ")
```

Gambar C. 4 Kode Sumber Fungsi `enterDeclarator` pada listener

```
enterSelectionstatement(ctx)
1   selectstat = ctx.getText()
2   array[] = selectstat.split("(")
3   seq = "H: " + array[0] + "("
5   ast.append(seq)
6   if array[0].contains("if")
7         CountLevel = CountLevel + 2
8         flagIf = 1
9   else if array[0].contains("switch")
10        Count_Level = Count_Level + 3
11        flagSwitchcase = 1
12  sequence.add(selectstat)
13  line.append(ctx.getStart().getLine())
14  level.append(count_level)

exitSelectionstatement(ctx)
1   selectstat = ctx.getText()
2   array[] = selectstat.split("(")
3   if array[0].contains("switch")
5         CountLevel = CountLevel - 3
6         flagSwitchcase = 0
7   else
8         CountLevel = CountLevel - 2
9         flagIf = 0
10  sequence = ") "
11  level.append(count_level)
12  ast.append(seq)
13  sequence.add("$")
14  line.append(-1)
```

Gambar C. 5 Kode Sumber Fungsi Selectionstatement pada Listener

```
enterIterationstatement(ctx)
1   iterstat = ctx.getText()
2   array[] = selectstat.split("(")
3   if array[0].equals("while")
4         seq= "I:while ( "
5   else if array[0].equals("for")
6         seq = "I:for ( "
7   else
8         flagIterDowhile = 1
9         sequence = "I:do ( "
10  level.append(count_level)
11  ast.append(seq)
12  sequence.add("$")
13  line.append(ctx.getStart().getLine())
14  CountLevel++
15  countIter = 1

exitIterationstatement(ctx)
1   flagIterDowhile = 0
2   sequence = ") "
3   CountLevel--
4   level.append(count_level)
5   ast.append(seq)
6   sequence.add("$")
7   line.append(-1)
```

Gambar C. 6 Kode Sumber Fungsi Iterationstatement pada Listener

```
enterCondition(ctx)
1   flagCondition = 1
2   seq = "( "
3   level.append(count_level)
4   ast.append(seq)
5   sequence.add("$")
6   line.append(-1)
7   CountLevel++
8   Condition[] = ctx.getText()
9   for i = 0 to size of condition
10        if     condition[i]     =     '<'     &&
          condition[i+1] = '='
11              seq = "J<= "
12        else   if   condition[i]   =   '<'   &&
          condition[i+1] != '='
13              seq = "J< "
14        else   if   (condition[i]   =   '>')   &&
          (condition[i-1]!= '=')
16              seq = "J> "
17        else if (condition[i] = '>') &&
          (condition[i-1]= '=')
18              seq = "J>= "
19        else   if   condition[i]   =   '='   &&
          condition [i+1] = '='
20              seq "J== "
21        else if condition[i] = '!'
                seq = "J!= "
22        else if condition[i] = '&'
23              seq = "J&& "
          else if condition[i] = '|'
24              seq = "J|| "
25        else   if   condition[i]   =   '+'   &&
          condition[i] != '+'
26              seq = "J+ "
27        else   if   condition[i]   =   '-'   &&
          condition[i+1] != '-'
28              seq = "J- "
29        else if condition[i] = '*'
                seq = "J* "
30        else if condition[i] = '/'
```

```
31              seq = "J/ "
32          else if condition[i] = '%'
33              seq = "J% "
34          level.append(count_level)
35          ast.append(seq)
36          sequence.add(ctx.getText())
37          line.append(ctx.getStart().getLine())

exitCondition(ctx)
1   flagCondition = 0
2   seq = ") "
3   CountLevel--
4   level.append(count_level)
5   ast.append(seq)
6   sequence.add("$")
7   line.append(-1)
```

Gambar C. 7 Kode Sumber Fungsi Condition pada listener

```
enterAssignmentexpression(ctx)
1   expression = ctx.getText().split("=")

exitAssignmentexpression(ctx)
1   if flag_express == 1 && flagparamcall == 0
2       flag_express = 0
3       sequence = ") "
4       count_Level--
5       level.append(count_level)
6       ast.append(seq)
7       sequence.add("$")
8       line.append(-1)
9   flag_postfix = 0
10  flag_primary =0
11  flag_string = 0
```

Gambar C. 8 Kode Sumber Fungsi Assignmentexpression pada listener

```
enterAssignmentoperator(ctx)
1   if flag_condition == 0
2         flag_express = 1
3         if flag_label > 0
4               operator = "G" + flag_label+
5               ctx.getText()
6         else
7               operator = ctx.getText()
8         if flag_postfix == -1
9               seq = ctx.getText()+"V" + +
10              postfix + " "
11              seq_ori = operator + "V" +
12              ctx.getText()
13        else if flag_postfix == -2
14              seq = operator + "this" +
16              temp_postfix + " "
17              seq_ori = operator + "this" +
18              temp_postfix + " "
19              postfix = null
20        else
21              seq = operator + "V "
22              seq_ori = ctx.getText() + "V "
23
24        level.append(count_level)
25        ast.append("( ")
26        sequence.add("$")
27        line.append(-1)
28        Count_level++
29        level.append(count_level)
30        ast.append(seq)
31        sequence.add(seq_ori)
32        line.append(ctx.getStart().getLine())
33        statement[] = ctx.getText().split("=")
34        for i = 0 to size of statement
35              if statement[i] = '+'
36                    seq = operator + "+ "
37              else if statement[i] = '-' &&
                  statement[i+1] != '>'
38                    seq = operator + "- "
```

```
39              else if (statement[i] = '/'
40                    seq = operator + "/ "
41              else if statement[i] = '%'
42                    seq = operator + "% "
43              else if statement[i] = '*'
44                    seq = operator + "* "
45              else if statement[i] = '"'
46                    break
47              level.append(count_level)
48              ast.append(seq)
49              sequence.add(ctx.getText())
50              line.append(ctx.getStart().getLin
                e())
```

Gambar C. 9 Kode Sumber Fungsi enterAssignmentoperator pada listener

```
enter enterUnqualifiedid(ctx)
1   if (flag_express == 1 || flagBrace == 1) &&
    flag_condition == 0
2         if flagparamcall > 0
3               if temp_param == null
4                     operator=operator+"P"
5               temp_param = "P"
6         else if flagparamcall == 0
7             operator=operator.replace("P","")
8         if postfix == null
9               if temp_postfix == null
10                    temp_postfix=null
11              else
12                    seq = operator+"V "
13                    level.append(count_level)
14                    ast.append(seq)
16                    sequence.add(operator+ctx.
                      getText())
17                    line.append(ctx.getStart()
                      .getLine())
18        else
19              if flag_postfix == 1
20                    temp_postfix = postfix
21              else if flag_postfix == 3
22                    temp_postfix = postfix
23                    flagparampost = 0
24                    flag_postfix = 4
25              seq = operator+"V"+postfix
26              level.append(count_level)
27              ast.append(seq)
28              sequence.add(operator+ctx.getTex
                t()+postfix)
29              line.append(ctx.getStart().getLi
                ne())
30              postfix == null
31  else if paramcall > 0 && flag_condition ==
    0
32              if flag_label > 0
33                    operator=      "G"      +
```

```
                    flag_label+"P"
34              else
35                  operator = "P"
36              if postfix == null
37                  if temp_postfix == null
38                      temp_postfix=null
39                  else
40                      seq = operator+"V "
41                      level.append(count_le
                        vel)
42                      ast.append(seq)
43                      sequence.add(operator
                        +ctx.getText())
44                      line.append(ctx.getSt
45                      art().getLine())
46              if flag_postfix == 1
47                  temp_postfix = postfix
48              else if flag_postfix == 3
49                  temp_postfix = postfix
50                  flagparampost = 0
51                  flag_postfix = 4
52              seq = operator+"V"+postfix
53              level.append(count_level)
54              ast.append(seq)
55              sequence.add(operator+ctx.getTex
                t()+postfix)
56              line.append(ctx.getStart().getLi
                ne())
57              postfix == null
58

59  else if flag_condition == 1
60          if flagparamcall > 0
61              operator = "JPV"
62          else
63              operator = "JPV"
64          if postfix == null
65              if temp_postfix == null
66                  temp_postfix=null
```

```
67                  else
68                        seq = operator+" "
69                        level.append(count_level)
70                        ast.append(seq)
71                        sequence.add(operator)
                          line.append(ctx.getStart()
72                        .getLine())
73
74            else
75                  if flag_postfix == 1
                          seq_ori   =   operator   +
                          ctx.getText() + postfix +
76                        " "
                          seq = operator + postfix
77                        + " "
78                        temp_postfix = postfix
79                  else if flag_postfix == 2
80                        flag_postfix = 5
                          seq_ori   =   operator   +
81                        ctx.getText() + " "
82                        seq = operator + " "
83                  else if flag_postfix == 3
84                        flag_postfix = 5
85                        seq_ori = ctx.getText()
                          seq = operator + postfix
86                        + " "
87                  else  if  flag_postfix  ==  5  ||
88                  flag_postfix == 6
89                        seq_ori = ctx.getText()
                          seq = operator + postfix
90                        + " "
91                  level.append(count_level)
92                  ast.append(seq)
93                  sequence.add(seq_ori)
94                  line.append(ctx.getStart().getLi
                    ne())
                    postfix = null
```

Gambar C. 10 Kode Sumber Fungsi enterUnqualifiedid pada Listener

```
enter enterLiteral(ctx)
1   if (flag_express == 1 || flagBrace == 1) &&
    flag_condition == 0
2         if flagparamcall > 0
3               if temp_param == null
4                     operator=operator+"P"
5               temp_param = "P"
6         else if flagparamcall == 0
7              operator=operator.replace("P","")
8         if flag_string == 1
9               seq = operator + "string"
10              seq_ori = operator + "string"
11        else
12              seq = operator + ctx.getText()
                + "string"
13              seq_ori     =     operator     +
                ctx.getText() + "string"
14        level.append(count_level)
16        ast.append(seq)
17        sequence.add(seq_ori)
18        line.append(ctx.getStart().getLine())
19  else if flagparamcall > 0 && flag_condition
    == 0
20        if flag_label > 0
21              operator = "G"+flag_label+"P"
22        else
23             operator = "P"
24
25        if flag_string == 1
26              seq = operator + "string"
27              seq_ori = operator + "string"
28        else
29              seq = operator + ctx.getText()
                + "string"
30              seq_ori     =     operator     +
                ctx.getText() + "string"
31        level.append(count_level)
32        ast.append(seq)
33        sequence.add(seq_ori)
```

```
34        line.append(ctx.getStart().getLine())
35  else if flag_condition == 1
36        if flagparamcall > 0
37              if flag_string == 1
38                    seq = "JPString"
39                    seq_ori  =  "JPString"  +
40                    ctx.getText()
                  else
41                    seq     =     "JP"     +
                      ctx.getText()
42                    seq_ori  =  "JPString"  +
                      ctx.getText()
43              level.append(count_level)
44              ast.append(seq)
45              sequence.add(seq_ori)
46              line.append(ctx.getStart().getLi
                ne())
47        else
48              seq = "J" + ctx.getText()
49              seq_ori = "J" + ctx.getText()
50              level.append(count_level)
51              ast.append(seq)
52              sequence.add(seq_ori)
53              line.append(ctx.getStart().getLi
                ne())
```

Gambar C. 11 Kode Sumber Fungsi enterLiteral pada Listener

```
enterPostfixexpression(ctx)
1   if (flagShift > 0 && flagShift2 = 1)
2         flagShift--
3   else
4         postfixexpres[] = ctx.getText()
5         flagString = 0
6         for i = 0 to size of postfixexpress
7               if postfixexpress[i]  = '"'
8                     flagString = 1
9                     break
10              else if postfixexpress[i] = '(
11                    break
```

Gambar C. 12 Kode Sumber Fungsi *enterPostfixexpression* bagian 1

```
1   if flagString = 0
2         if (regexPattern2_1 = match) ||
3            (regexPattern2_2 = match)
4               flagPostfix = 2
5               if (flagSwitchcase > 0 &&
                (flagCondition=0) &&
                (flagExpression!=1) &&
                (flagParametercall = 0)
6                     sequence = "G" +
7                     flagSwitchcase + "VO"
8                     ast.append(sequence)
9                     level.append(CountLevel)
10              else if (flagExpression!=1) &&
                (flagCondition = 0) &&
                (flagParametercall = 0)
11                    sequence = "VO"
12                    ast.append(sequence)
13                    level.append(CountLevel)
14              else
15                    postfix = "O"
```

Gambar C. 13 Kode Sumber Fungsi *enterPostfixexpression* bagian 2

```
1   if flagString = 0
2         if (regexPattern3_11 = match) ||
3            (regexPattern3_22 = match)
4               flagpostfix = 3
5               Postfix = "->O"
                if (flagSwitchcase > 0 &&
                (flagCondition=0) &&
                (flagExpression!=1) &&
6               (flagParametercall = 0)
7                     sequence = "G" +
8                     flagSwitchcase + "V->O"
9                     ast.append(sequence)
10                    level.append(CountLevel)
                else if (flagExpression!=1) &&
                (flagCondition = 0) &&
11              (flagParametercall = 0)
12                    sequence = "V->O"
13                    ast.append(sequence)
14                    level.append(CountLevel)
15                    postfix = NULL
16              else if flagParametercall = 1
17                    flagParameterpostfix = 1
```

Gambar C. 14 Kode Sumber Fungsi `enterPostfixexpression` bagian 3

```
1  else if regexPattern11 = match
2     if (flagPostfix !=3 ||
      flagParametercall=1)&&flagParameterpostfix
      =0
3              flagPostfix = 1
4              postfix = "."
```

Gambar C. 15 Kode Sumber Fungsi `enterPostfixexpression` bagian 3

```
1   else if postfixexpress.contains("--")
2         if flagSwitchcase>0&&flagCondition=0
                sequence = "G" + flagSwitchcase
3               + "V--"
                ast.append(sequence)
4               level.append(CountLevel)
5               postfix = NULL
6         else if flagCondition = 0
7               sequence = "V--"
8               ast.append(sequence)
9               level.append(CountLevel)
10        else if flagCondition = 1
11              postfix = "--"
12              flagPostfix = 6
13
14  else if postfixexpress contains "++"
15        if flagSwitchcase>0&&flagCondition=0
16              sequence = "G" + flagSwitchcase
17              + "V++"
                ast.append(sequence)
18              level.append(CountLevel)
                postfix = NULL
19        else if flagCondition = 0
20              sequence = "V++"
21              ast.append(sequence)
22              level.append(CountLevel)
23        else if flagCondition = 1
24              postfix = "++"
25              flagPostfix = 5
```

Gambar C. 16 Kode Sumber Fungsi enterPostfixexpression bagian 4

```
Smithwaterman(sequence1, sequence2, level1,
level2)
 1   row = firstCodeSequence.size()
 2   coloumn = secondCodeSequence.size()
 3   w = -1
 4   match = 2
```

```
5   mismatch  = -1
6   for a = 0 to row
7       H[a][0] = 0
8   for a = 0 to col
9       H[0][a] = 0
10  H[0][0] = 0
11  for i = 1 to row
12      for j = 1 to col
13         max  = 0
14         if max  == 0
15             if secondCodeSequence.get(j-1).equals(firstCodeSequence.get(i-1)) && secondLevel.get(j-1).equals(firstLevel.get(i-1))
16                 max = H[i-1][j-1] + match
17                 direction[i-1][j-1] = 1
18             else
19                 max= H[i-1][j-1]+mismatch
20                 direction[i-1][j-1] = 9
21          if max < (H[i][j-1] + w))
22              //insertion atas
23              max = H[i][j-1] + w
24              direction[i-1][j-1] = 3
25          if max < (H[i-1][j] + w))
26              //deletion kiri
27              max = H[i-1][j] + w
28              direction[i-1][j-1] = 2
29          H[i][j] = max
30  //call smith waterman's backtracking
31  jumlah = backtrack() //Kode Sumber 4.29
```

Gambar C. 17 Kode Sumber Metode Smith-Waterman

```
Smith_modif(sequence1, sequence2, level1,
level2)
 1   row = firstCodeSequence.size()
 2   coloumn = secondCodeSequence.size()
 3   w = -1
 4   match = 2
 5   mismatch  = -1
 6   for a = 0 to row
 7      H_modif[a][0] = 0
 8      check_Row[a] = 0
 9   for a = 0 to col
10      check_Col[a] = 0
11      H_modif[0][a] = 0
12   H[0][0] = 0
13   for i = 1 to row
14      for j = 1 to col
15         max  = 0
16         if max  == 0
                 if secondCodeSequence.get(j-
                 1).equals(firstCodeSequence.get
17               (i-1)) && secondLevel.get(j-
                 1).equals(firstLevel.get(i-1))
                 && check_Col[j-1] == 0 &&
                 check_Row[i-1]==0
                    max = H_modif[i-1][j-1] +
18                  match
19                  direction_modif[i-1][j-1]=1
20                  check_Col[j-1] = -1
21                  check_Row[i-1] = -1

                    index.add((i-1) + "," + (j-
22                  1))
23               else
```

```
                    max = H_modif[i-1][j-1] +
24                  mismatch
                    direction_modif[i-1][j-1] =
25                  9
26          if max < (H[i][j-1] + w))
27                  //insertion atas
28                  max = H_modif[i][j-1] + w
29                  direction_modif[i-1][j-1] = 3
30          if max < (H_modif[i-1][j] + w))
31                  //deletion kiri
32                  max = H_modif[i-1][j] + w
33                  direction_modif[i-1][j-1] = 2
34                  H_modif[i][j] = max
35  backtrack_modif() // Kode Sumber 4.31
```

Gambar C. 18 Kode Sumber Metode Smith-Waterman Modifikasi

```
Int Backtrack()
 1    k = row, l = coloumn, c = 0, b=0
 2    while k >= 0 && l >= 0
 3          if direction[k][l] == 2
 4                  //deletion --> kiri
 5                  k = k-1
 6                  l = l
 7          else if direction[k][l] == 3
 8                   //insertion --> katas
 9                  k = k
10                  l = l-1
11          else
12            if direction[k][l] ==  1
13                  hasil[jumlah] =
```

```
                    secondCodeSequence.get(l)
 14                 res[jumlah] = l-1
                    index_smith.add((k) + "," +
 15                 (l))
 16                 jumlah++
 17                 k--
 18                 l--
 19   return jumlah
double similarity()
  1   if row > col
  2           total = row
  3   else
  4           total = col
  5   persentase  = 100.0 * jumlah / total
  6   return persentase
```

Gambar C. 19 Kode Sumber Backtracking dan Similarity Metode Smith-Waterman

```
  1  maks = row, val = index.size()
  2  hs_i.clear()
  3  for v = 0 to maks
  4        if index.size()==0
  5              break
  6              y = v
  7              flag = 0
  8         for x = 0 to index.size()
  9                ind = index.size(x)
                   ind_i =
                   Integer.parseInt(ind.split(","
 10                )[0])
                   ind_ j=
 11                Integer.parseInt(ind.split(","
```

```
                    )[1])
12                  if ind_i == v
13                        flag = 1
14                        hs_i.add(-1)
15                        while y > -1
                              if
                              hs_i.get(y).equals(i
16                            nd_j))
17                                  ct = 0
18                                  break
                              else if
                              !hs_i.get(y).equals(
19                            ind_j))
20                                  ct = 1
21                                  y--
22                        if ct == 1
23                              hs_i.remove(v)
24                              hs_i.add(v, ind_j)
25                              index.remove(x)
26                              break
27            if flag == 0
28                  hs_i.add(-1)
29  simi_modif = 100.0 * val / row
```

Gambar C. 20 Kode Sumber Backtracking dan Similarity Metode Smith-Waterman yang dimodifikasi

BIODATA PENULIS

Rachmania Ilavi dilahirkan di Surabaya pada tanggal 9 September 1994. Penulis merupakan anak kedua dari pasangan Bapak Sugeng Hery Sucipto dan Ibu Suliarsih. Penulis menempuh pendidikan formal dimulai dari TK Khadijah Surabaya (1999-2000), SD Khadijah Surabaya (2000-2006), SMP Negeri 12 Surabaya (2006-2009), SMA Negeri 5 Surabaya (2009-2012), dan S1 Teknik Informatika ITS (2012-2016). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Algoritma dan Pemrograman (AP). Penulis aktif dalam organisasi seperti Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) pada tahun 2013-2015. Penulis juga aktif dalam berbagai kegiatan kepanitiaan yaitu SCHEMATICS 2013 divisi KESTARI dan SCHEMATICS 2014 divisi REEVA. Penulis mempunyai hobi *travelling*, menulis cerita pendek, dan mendengarkan musik. Penulis dapat dihubungi melalui email: rachmaniailavi@gmail.com